

LNCs 4988

Rudolf Berghammer
Bernhard Möller
Georg Struth (Eds.)

Relations and Kleene Algebra in Computer Science

10th International Conference on Relational
Methods in Computer Science
and 5th International Conference on Applications
of Kleene Algebra, RelMiCS/AKA 2008
Frauenwörth, Germany, April 2008, Proceedings

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Rudolf Berghammer Bernhard Möller
Georg Struth (Eds.)

Relations and Kleene Algebra in Computer Science

10th International Conference on Relational
Methods in Computer Science
and 5th International Conference on Applications
of Kleene Algebra, RelMiCS/AKA 2008
Frauenwörth, Germany, April 7-11, 2008
Proceedings

Volume Editors

Rudolf Berghammer

Christian-Albrechts-Universität zu Kiel, Institut für Informatik

Olshausenstraße 40, 24098 Kiel, Germany

E-mail: rub@informatik.uni-kiel.de

Bernhard Möller

Universität Augsburg, Institut für Informatik

Universitätsstr. 14, 86135 Augsburg, Germany

E-mail: bernhard.moeller@informatik.uni-augsburg.de

Georg Struth

University of Sheffield, Department of Computer Science

Regent Court, 211 Portobello, Sheffield S1 4DP, UK

E-mail: g.struth@dcs.shef.ac.uk

Library of Congress Control Number: 2008923359

CR Subject Classification (1998): F.4, I.1, I.2.3, D.2.4

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743

ISBN-10 3-540-78912-X Springer Berlin Heidelberg New York

ISBN-13 978-3-540-78912-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2008

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper SPIN: 12249879 06/3180 5 4 3 2 1 0

Preface

This volume contains the proceedings of the 10th International Seminar on Relational Methods in Computer Science (RelMiCS 10) and the 5th International Workshop on Applications of Kleene Algebra (AKA 5). The joint conference took place in Frauenwörth on an Island in Lake Chiem in Bavaria, April 7–April 11, 2008. Its purpose was to bring together researchers various subdisciplines of computer science, mathematics and related fields who use the calculus of relations and/or Kleene algebra as methodological and conceptual tools in their work.

This conference is the joint continuation of two different strands of meetings. The seminars of the RelMiCS series were held in Schloss Dagstuhl (Germany) in January 1994, Parati (Brazil) in July 1995, Hammamet (Tunisia) in January 1997, Warsaw (Poland) in September 1998, Québec (Canada) in January 2000, and Oisterwijk (The Netherlands) in October 2001. The meeting on Applications of Kleene Algebra started as a workshop, also held in Schloss Dagstuhl, in February 2001. To join these two themes in one conference was mainly motivated by the substantial common interests and overlap of the two communities. Over the years this has led to fruitful interactions and opened new and interesting research directions. Joint meetings have been held in Malente (Germany) in May 2003, in St. Catherines (Canada) in February 2005 and in Manchester (UK) in August/September 2006.

This volume contains 28 contributions by researchers from all over the world. Next to 26 regular papers there were the invited talks “Formal Methods and the Theory of Social Choice” by Marc Pauly (Stanford University, USA) and “Relations Making Their Way from Logics to Mathematics and Applied Sciences” by Gunther Schmidt (University of the Armed Forces Munich, Germany). The papers show that relational and Kleene algebra methods have wide-ranging diversity and applicability in theory and practice.

In addition, for the second time, a PhD programme was offered. It included the invited tutorials “Basics of Relation Algebra” by Jules Desharnais (Université Laval, Québec, Canada), “Basics of Modal Kleene Algebra” by Georg Struth (University of Sheffield, UK) and “Applications to Preference Systems” by Susanne Saminger (Universität Linz, Austria).

We are very grateful to the members of the Programme Committee and the external referees for their care and diligence in reviewing the submitted papers. We also want to thank Roland Glück, Peter Höfner Iris Kellner and Ulrike Pollakowski for their assistance; they made organizing this meeting a pleasant experience. We also gratefully appreciate the excellent facilities offered by the EasyChair conference administration system. Finally, we want to thank our sponsors ARIVA.DE AG (Kiel), CrossSoft (Kiel), HSH Nordbank AG (Kiel) and the Deutsche Forschungsgemeinschaft DFG for their financial support.

April 2008

Rudolf Berghammer
Bernhard Möller
Georg Struth

Organization

Programme Committee

R. Berghammer	Kiel, Germany
H. de Swart	Tilburg, The Netherlands
J. Desharnais	Laval, Canada
M. Frías	Buenos Aires, Argentina
H. Furusawa	Kagoshima, Japan
P. Jipsen	Chapman, USA
W. Kahl	McMaster, Canada
Y. Kawahara	Kyushu, Japan
B. Möller	Augsburg, Germany
C. Morgan	Sydney, Australia
M. Ojeda Aciego	Málaga, Spain
E. Orłowska	Warsaw, Poland
S. Saminger	Linz, Austria
G. Schmidt	Munich, Germany
R. Schmidt	Manchester, UK
G. Scollo	Catania, Italy
A. Szalas	Linköping, Sweden
G. Struth	Sheffield, UK
J. van Benthem	Amsterdam, The Netherlands
M. Winter	Brock, Canada

External Referees

Natasha Alechina	Peter Höfner
Bernd Braßel	Britta Kehden
Domenico Cantone	David Rydeheard
Patrik Eklund	Dmitry Tishkovsky
Alexander Fronk	Dimiter Vakarelov
Joanna Golinska-Pilarek	

Table of Contents

Formal Methods and the Theory of Social Choice (Invited Talk)	1
<i>Marc Pauly</i>	
Relations Making Their Way from Logics to Mathematics and Applied Sciences (Invited Talk)	3
<i>Gunther Schmidt</i>	
Boolean Logics with Relations	4
<i>Philippe Balbiani and Tinko Tinchev</i>	
Relation Algebra and RELVIEW in Practical Use: Construction of Special University Timetables	22
<i>Rudolf Berghammer and Britta Kehden</i>	
A Relation Algebraic Semantics for a Lazy Functional Logic Language	37
<i>Bernd Braßel and Jan Christiansen</i>	
Latest News about Demonic Algebra with Domain	54
<i>Jean-Lou De Carufel and Jules Desharnais</i>	
On the Structure of Demonic Refinement Algebras with Enabledness and Termination	69
<i>Jean-Lou De Carufel and Jules Desharnais</i>	
Multi-objective Problems in Terms of Relational Algebra	84
<i>Florian Diedrich, Britta Kehden, and Frank Neumann</i>	
The Lattice of Contact Relations on a Boolean Algebra	99
<i>Ivo Düntsch and Michael Winter</i>	
A Non-probabilistic Relational Model of Probabilistic Kleene Algebras	110
<i>Hitoshi Furusawa, Norihiro Tsumagari, and Koki Nishizawa</i>	
Increasing Bisemigroups and Algebraic Routing	123
<i>Timothy G. Griffin and Alexander J.T. Gurney</i>	
Lazy Relations	138
<i>Walter Guttmann</i>	
The Algebraic Approach I: The Algebraization of the Chomsky Hierarchy	155
<i>Mark Hopkins</i>	

The Algebraic Approach II: Dioids, Quantales and Monads	173
<i>Mark Hopkins</i>	
Automated Reasoning for Hybrid Systems — Two Case Studies	191
<i>Peter Höfner</i>	
Non-termination in Idempotent Semirings	206
<i>Peter Höfner and Georg Struth</i>	
Formal Concepts in Dedekind Categories	221
<i>Toshikazu Ishida, Kazumasa Honda, and Yasuo Kawahara</i>	
The Structure of the One-Generated Free Domain Semiring	234
<i>Peter Jipsen and Georg Struth</i>	
Determinisation of Relational Substitutions in Ordered Categories with Domain	243
<i>Wolfram Kahl</i>	
Boolean Algebras and Stone Maps in Schröder Categories	259
<i>Yasuo Kawahara</i>	
Cardinality in Allegories	274
<i>Yasuo Kawahara and Michael Winter</i>	
Solving Linear Equations in $\ast$ -continuous Action Lattices	289
<i>Béchir Ktari, François Lajeunesse-Robert, and Claude Bolduc</i>	
Reactive Probabilistic Programs and Refinement Algebra	304
<i>L.A. Meinicke and K. Solin</i>	
Knowledge and Games in Modal Semirings	320
<i>Bernhard Möller</i>	
Theorem Proving Modulo Based on Boolean Equational Procedures	337
<i>Camilo Rocha and José Meseguer</i>	
Rectangles, Fringes, and Inverses	352
<i>Gunther Schmidt</i>	
An Ordered Category of Processes	367
<i>Michael Winter</i>	
Automatic Proof Generation in Kleene Algebra	382
<i>James Worthington</i>	
Author Index	397

Formal Methods and the Theory of Social Choice

Marc Pauly

Department of Philosophy, Stanford University

Social Choice Theory

Social Choice Theory (SCT, see [2] for an introduction) studies social aggregation problems, i.e., the problem of aggregating individual choices, preferences, opinions, judgments, etc. into a group choice, preference, opinion or judgment. Examples of such aggregation problems include the following: aggregating the political opinions of a country's population in order to choose a president or parliament, assigning college students to dormitories based on their preferences, dividing an inheritance among a number of people, and matching romance-seeking web users at an internet dating site. On the one hand, SCT analyzes existing aggregation mechanisms, e.g. the voting procedures of different countries or different matching algorithms. On the other hand, SCT explores different normative properties such as anonymity or neutrality, and the logical dependencies among them. The central results in SCT fall into the second category, the most well-known being Arrow's impossibility theorem [1] and the Gibbard-Satterthwaite theorem [3,8].

When social choice theorists talk about the link between SCT and logic, they usually refer to results like Arrow's theorem. It is a result using logic in the sense that it shows that a number of (prima facie) natural and desirable conditions that can be imposed on a voting procedure are inconsistent when taken together. The logician, however, would point out that the use of logic in these results is restricted to the kind of logic that is used in much mathematical reasoning. It is only more recently that formal logic and formal methods more generally have been introduced to social choice theory. In this talk, I will argue that this is a fruitful avenue of research by giving two examples of these new contacts between SCT and formal methods.

Formal Methods

What is needed in order to apply formal methods to SCT is to take a more formal approach to the language, axioms and theorems of SCT. The key step here is the introduction of formal languages. Once we have formulated the axioms and theorems of SCT in a formal language, various meta-theoretic questions can be asked about SCT. In fact, the step from SCT to meta-SCT is analogous to the step from mathematics to meta-mathematics. It allows us to ask questions about axiomatizability, definability, decidability, etc. that are typical benefits of the formal approach. This methodological view has been argued for in [6]. In this talk, I will give two examples of results that can be obtained in this approach, one example that provides a new characterization of majority voting, and a second example that looks at how much of social choice theory can be carried out in first-order logic.

SCT knows different axiomatic characterizations of majority voting. The most famous result by May [4] states that a voting procedure satisfies anonymity, neutrality and positive responsiveness if and only if it is the majority rule. The new characterization using the methods of formal logic captures majority voting using axioms formulated in a particular logical language. These results are reported in [5]. As a second example, we consider SCT as a first-order theory, the theory of multiple linear orders over a set of alternatives. We can look at what voting procedures and normative properties are definable in such a framework. Furthermore, we can study whether such a first-order theory is decidable. The formal details of this approach are outlined in [7].

References

1. Arrow, K.: Social Choice and Individual Values. Yale University Press, New Haven, London (1951)
2. Gaertner, W.: A Primer in Social Choice Theory. Oxford University Press, Oxford (2006)
3. Gibbard, A.: Manipulation of voting schemes: A general result. *Econometrica* 41, 587–601 (1973)
4. May, K.O.: A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica* 20, 680–684 (1952)
5. Pauly, M.: Axiomatizing collective judgment sets in a minimal logical language. *Synthese* 158, 233–250 (2007)
6. Pauly, M.: On the role of language in social choice theory. *Synthese* (to appear)
7. Pauly, M.: Social Choice in First-Order Logic: Investigating Decidability & Definability (unpublished)
8. Satterthwaite, M.: Strategy-proofness and arrow’s conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory* 10, 187–217 (1975)

Relations Making Their Way from Logics to Mathematics and Applied Sciences

Invited Lecture

Gunther Schmidt

Institute for Software Technology, Department of Computing Science
Universität der Bundeswehr München, 85577 Neubiberg, Germany
`Gunther.Schmidt@unibw.de`

The study of relations emerged within the realm of (Algebraic) Logics around the 1850s. At that time, computers were not yet in existence, nor did there exist programming languages or semantics to interpret them. Matrices did come into common use only a hundred years later. Not even the theory of sets had been fully developed. As a consequence, relations carry with them quite a burden of historic presentation. Even in these days, texts appear containing a detailed exegesis of Schröder's work.

Today, however, we may also observe that relations are increasingly used in other fields, first in mathematics, but in the meantime also in engineering and social sciences. A pre-requisite for broader use was the transition to heterogeneous relations together with a discipline of typing — as opposed to working with the unwieldy universe containing everything. One will now start with sets and relations as small as possible derived from the application contexts and construct what is needed in a generically sound way. To be easily comprehensible, this requires not least pointfreeness.

In Mathematics, the Homomorphism and Isomorphism Theorems have been reworked and presented at RelMiCS 9 in Manchester. In the meantime, aspects of topology, closure forming, and lattices acquired more and more relational flavour. Among the examples to be presented from other application areas are those in system dynamics, in social choice functions, or just in Sudoku solving. It will be mentioned where German trade unions work with relations and continuously refer to relational papers of our circle.

Boolean Logics with Relations

Philippe Balbiani¹ and Tinko Tinchev²

¹ Institut de Recherche en Informatique de Toulouse, Toulouse University, France

² Faculty of Mathematics and Computer Science, Sofia University, Bulgaria

Abstract. We study a fragment of propositional modal logics using the universal modality given by a restriction on the modal depth of modal formulas.

Keywords: First-order classical logic, propositional modal logic, Boolean algebra, relations.

1 Introduction

Modal languages are usually considered as expressive languages for talking about relational structures. There is an important literature concerning the correspondence theory, the decidability/complexity and the axiomatization/completeness of various fragments of propositional modal logics obtained when their languages are restricted somehow or other [3,8,11]. In a number of disciplines of artificial intelligence and theoretical computer science, properties of artificial agents and computer programs essentially amount to safety properties and liveness properties. Safety properties can be expressed by modal formulas of the form $[U](start \wedge \phi \rightarrow \Box(end \rightarrow \psi))$ (“if ϕ holds upon the start of an execution then if this execution terminates then ψ holds upon termination”) whereas liveness properties can be expressed by modal formulas of the form $[U](start \wedge \phi \rightarrow \Diamond(end \wedge \psi))$ (“if ϕ holds upon the start of an execution then this execution terminates and ψ holds upon termination”). In these formulas, $[U]$ means “at all time points”, $\Box$ means “at every time point after the reference point” and $\Diamond$ means “at some time point after the reference point”. Moreover, ϕ and ψ denote respectively a precondition and a postcondition. In most cases, preconditions and postconditions contain no modal operators. Thus, an obvious question is why we define languages of modal logic in the form of a general rule like $\phi ::= a \mid \perp \mid \neg\phi \mid (\phi_1 \vee \phi_2) \mid \Box\phi \mid [U]\phi$, where a denotes a Boolean term and not in the form of a restricted rule like $\phi ::= [U](a_1 \rightarrow \Box a_2) \mid [U](a_1 \rightarrow \Diamond a_2) \mid \perp \mid \neg\phi \mid (\phi_1 \vee \phi_2)$, where a_1 and a_2 denote Boolean terms. To give evidence that such a restriction is fruitful, let us focus here on the following modal formulas:

- $[U](x \rightarrow \Diamond x)$,
- $[U](x \rightarrow \Box\neg y) \rightarrow [U](y \rightarrow \Box\neg x)$,
- $[U](x \rightarrow \Diamond z) \wedge [U](z \rightarrow \Diamond y) \rightarrow [U](x \rightarrow \Diamond y)$,

where x , y and z denote Boolean variables. It is easy to verify that their standard translations in the language of first-order logic are respectively equivalent to the following first-order formulas:

- $\forall s(R(s, s))$,
- $\forall s\forall t(R(s, t) \rightarrow R(t, s))$,
- $\forall s\forall t(\exists u(R(s, u) \wedge R(u, t)) \rightarrow R(s, t))$.

This remark gives us a new research agenda for investigating the correspondence theory, the decidability/complexity and the axiomatization/completeness of fragments of propositional modal logics using the universal modality given by restrictions on the modal depth of modal formulas similar to the restriction suggested by the above rule. Due to space limitation, only fragments similar to the one given by the following restricted rule will be considered: $\phi ::= [U](a_1 \rightarrow \Box a_2) \mid [U](a \rightarrow \Diamond \perp) \mid \perp \mid \neg\phi \mid (\phi_1 \vee \phi_2)$. These fragments will be called “Boolean logics with relations” for reasons that will become obvious during the course of the paper. Section 2 introduces their syntax. Their two semantics are given in sections 3.1 and 3.2. The first semantics is based on the notion of Kripke frame whereas the second semantics is based on the notion of Boolean frame. Section 4 examines our restricted modal language as a tool for talking about Kripke frames and Boolean frames. It initiates the study of its correspondence theory. The decidability/complexity issue and the axiomatization/completeness issue are addressed in sections 5 and 6. In section 7, the concepts of weak canonicity and strong canonicity are introduced.

2 Syntax

We now set up the Boolean logic with relations as a modal language. Let $\mathcal{R}$ be a countably infinite set of relation symbols denoted by capital Latin letters P , Q , etc, possibly with subscripts. Each P in $\mathcal{R}$ is assumed to be n -placed for some integer $n \geq 0$ depending on P . To formalize the language $L_{\mathcal{R}}$, we need the following logical symbols: (1) symbols denoted by the letters (and) (parentheses), (2) a symbol denoted by the letter , (comma), (3) a countably infinite set of Boolean variables denoted by lower case Latin letters x , y , etc, possibly with subscripts, (4) Boolean functions 0 , $-$ and $\cup$, (5) a symbol denoted by the letter $\equiv$ and (6) Boolean connectives $\perp$, $\neg$ and $\vee$. We assume that no relation symbol in $\mathcal{R}$ occurs in the above list. Certain strings of logical symbols, called Boolean terms, will be denoted by lower case Latin letters a , b , etc, possibly with subscripts. They are defined by the following rule:

- $a ::= x \mid 0 \mid -a \mid (a_1 \cup a_2)$.

A Boolean term of the form x or $-x$ is called a Boolean literal. The modal formulas of $L_{\mathcal{R}}$ will be denoted by lower case Greek letters ϕ , ψ , etc, possibly with subscripts. They are defined by the following rule:

- $\phi ::= P(a_1, \dots, a_n) \mid a_1 \equiv a_2 \mid \perp \mid \neg\phi \mid (\phi_1 \vee \phi_2)$.

Thus, the similarity type of the language $L_{\mathcal{R}}$ is the structure $\tau = \langle \mathcal{R}, \rho \rangle$ where ρ is an arity function mapping the relation symbols P of $\mathcal{R}$ to appropriate integers $\rho(P) \geq 0$. In the above rule, note that we require that $\rho(P) = n$. Let us adopt the standard rules for omission of the parentheses. We define the other constructs as usual. In particular: 1 is -0 , $(a_1 \cap a_2)$ is $-(-a_1 \cup -a_2)$, $\top$ is $\neg \perp$ and $(\phi_1 \wedge \phi_2)$ is $\neg(\neg\phi_1 \vee \neg\phi_2)$. We use $\phi(x_1, \dots, x_n)$ to denote a modal formula whose Boolean variables form a subset of $\{x_1, \dots, x_n\}$. In this case, $\phi(a_1, \dots, a_n)$ will denote the modal formula obtained from $\phi(x_1, \dots, x_n)$ by simultaneously and uniformly substituting the Boolean terms $a_1, \dots, a_n$ for the Boolean variables $x_1, \dots, x_n$. For all sets Δ of modal formulas, we use $BV(\Delta)$ to denote the set of all Boolean variables occurring in Δ . Similarly, we use $BV(a)$ to denote the set of all Boolean variables occurring in the Boolean term a and we use $BV(\phi)$ to denote the set of all Boolean variables occurring in the formula ϕ .

3 Semantics

3.1 Kripke Semantics

A Kripke frame for $L_{\mathcal{R}}$ is a structure $\mathcal{F} = \langle S, I \rangle$ where S is a nonempty set and I is an interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I(P)$ on S . A valuation on $\mathcal{F}$ is an interpretation function V mapping the Boolean variables to subsets of S . We inductively define the interpretation function $\tilde{V}$ mapping the Boolean terms to subsets of S as follows:

- $\tilde{V}(x) = V(x)$,
- $\tilde{V}(0) = \emptyset$,
- $\tilde{V}(-a) = S \setminus \tilde{V}(a)$,
- $\tilde{V}(a_1 \cup a_2) = \tilde{V}(a_1) \cup \tilde{V}(a_2)$.

A Kripke model for $L_{\mathcal{R}}$ is a structure $\mathcal{M} = \langle \mathcal{F}, V \rangle$ where $\mathcal{F} = \langle S, I \rangle$ is a Kripke frame for $L_{\mathcal{R}}$ and V is a valuation on $\mathcal{F}$. We inductively define the notion of a modal formula ϕ being true in a Kripke model $\mathcal{M} = \langle S, I, V \rangle$, in symbols $\mathcal{M} \models \phi$, as follows:

- $\mathcal{M} \models P(a_1, \dots, a_n)$ iff there exist s_1 in $\tilde{V}(a_1)$, $\dots$, there exist s_n in $\tilde{V}(a_n)$ such that $(s_1, \dots, s_n) \in I(P)$,
- $\mathcal{M} \models a_1 \equiv a_2$ iff $\tilde{V}(a_1) = \tilde{V}(a_2)$,
- $\mathcal{M} \not\models \perp$,
- $\mathcal{M} \models \neg\phi$ iff $\mathcal{M} \not\models \phi$,
- $\mathcal{M} \models \phi_1 \vee \phi_2$ iff $\mathcal{M} \models \phi_1$ or $\mathcal{M} \models \phi_2$.

It follows from this definition that for all binary relations symbols P , if one interprets $\Box$ and $\Diamond$ by means of $I(P)$ then $\neg P(a_1, -a_2)$ is equivalent to $[U](a_1 \rightarrow \Box a_2)$ and $a \equiv 0$ is equivalent to $[U](a \rightarrow \Diamond \perp)$. The following modal formulas are true in all Kripke models:

- $P(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n) \rightarrow a_i \neq 0$,
- $P(a_1, \dots, a_{i-1}, (a'_i \cup a''_i), a_{i+1}, \dots, a_n) \leftrightarrow (P(a_1, \dots, a_{i-1}, a'_i, a_{i+1}, \dots, a_n) \vee P(a_1, \dots, a_{i-1}, a''_i, a_{i+1}, \dots, a_n))$.

A set Σ of modal formulas is said to be satisfiable in a Kripke frame $\mathcal{F} = \langle S, I \rangle$, in symbols $\mathcal{F} \text{ sat } \Sigma$, iff there exists a Kripke model $\mathcal{M} = \langle S, I, V \rangle$ based on $\mathcal{F}$ such that all modal formulas in Σ are true in $\mathcal{M}$. We shall say that a set Σ of modal formulas is satisfiable in a class $\mathcal{C}$ of Kripke frames, in symbols $\mathcal{C} \text{ sat } \Sigma$, iff Σ is satisfiable in some Kripke frame in $\mathcal{C}$. A modal formula ϕ is said to be a valid consequence of a set Σ of modal formulas in a Kripke frame $\mathcal{F} = \langle S, I \rangle$, in symbols $\Sigma \Vdash_{\mathcal{F}} \phi$, iff for all Kripke models $\mathcal{M} = \langle S, I, V \rangle$ based on $\mathcal{F}$, if all modal formulas in Σ are true in $\mathcal{M}$ then ϕ is true in $\mathcal{M}$. We shall say that a modal formula ϕ is a valid consequence of a set Σ of modal formulas in a class $\mathcal{C}$ of Kripke frames, in symbols $\Sigma \Vdash_{\mathcal{C}} \phi$, iff ϕ is a valid consequence of Σ in all Kripke frames in $\mathcal{C}$. For all sets Φ of modal formulas, $\mathcal{C}_{\Phi}^K$ will denote the class of all Kripke frames on which Φ is valid.

Proposition 1. *Let Φ , Σ be sets of modal formulas and ϕ be a modal formula such that $\Sigma \Vdash_{\mathcal{C}_{\Phi}^K} \phi$. If $BV(\Sigma)$ is finite then there exists a finite subset Σ' of Σ such that $\Sigma' \Vdash_{\mathcal{C}_{\Phi}^K} \phi$.*

Proof. Assume $BV(\Sigma)$ is finite. Consequently, there exist finitely many logically different modal formulas in $BV(\Sigma)$. Hence, there exists a finite subset Σ' of Σ such that $\Sigma' \Vdash_{\mathcal{C}_{\Phi}^K} \phi$.

Proposition 2. *Let $\mathcal{C}$ be a class of Kripke frames, Σ be a set of modal formulas and ϕ , ψ be modal formulas such that $\Sigma \cup \{\phi\} \Vdash_{\mathcal{C}} \psi$. Then $\Sigma \Vdash_{\mathcal{C}} \phi \rightarrow \psi$.*

Proof. The proposition directly follows from the definition of $\Vdash_{\mathcal{C}}$.

3.2 Boolean Semantics

A Boolean frame for $L_{\mathcal{R}}$ is a structure $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ where $\langle A, 0_A, -_A, \cup_A \rangle$ is a nondegenerate Boolean algebra and I is an interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I(P)$ on A such that

- for all $a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n$ in A , if $(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n) \in I(P)$ then $a_i \neq 0_A$,
- for all $a_1, \dots, a_{i-1}, a'_i, a''_i, a_{i+1}, \dots, a_n$ in A , $(a_1, \dots, a_{i-1}, a'_i \cup_A a''_i, a_{i+1}, \dots, a_n) \in I(P)$ iff $(a_1, \dots, a_{i-1}, a'_i, a_{i+1}, \dots, a_n) \in I(P)$ or $(a_1, \dots, a_{i-1}, a''_i, a_{i+1}, \dots, a_n) \in I(P)$.

A valuation on $\mathcal{F}$ is an interpretation function V mapping the Boolean variables to elements of A . We inductively define the interpretation function $\tilde{V}$ mapping the Boolean terms to elements of A as follows:

- $\tilde{V}(x) = V(x)$,
- $\tilde{V}(0) = 0_A$,
- $\tilde{V}(-a) = -_A \tilde{V}(a)$,
- $\tilde{V}(a_1 \cup a_2) = \tilde{V}(a_1) \cup_A \tilde{V}(a_2)$.

A Boolean model for $L_{\mathcal{R}}$ is a structure $\mathcal{M} = \langle \mathcal{F}, V \rangle$ where $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ is a Boolean frame for $L_{\mathcal{R}}$ and V is a valuation on $\mathcal{F}$. We inductively define the notion of a modal formula ϕ being true in a Boolean model $\mathcal{M} = \langle A, 0_A, -_A, \cup_A, I, V \rangle$, in symbols $\mathcal{M} \models \phi$, as follows:

- $\mathcal{M} \models P(a_1, \dots, a_n)$ iff $(\tilde{V}(a_1), \dots, \tilde{V}(a_n)) \in I(P)$,
- $\mathcal{M} \models a_1 \equiv a_2$ iff $\tilde{V}(a_1) = \tilde{V}(a_2)$,
- $\mathcal{M} \not\models \perp$,
- $\mathcal{M} \models \neg\phi$ iff $\mathcal{M} \not\models \phi$,
- $\mathcal{M} \models \phi_1 \vee \phi_2$ iff $\mathcal{M} \models \phi_1$ or $\mathcal{M} \models \phi_2$.

It follows from this definition that our Boolean models are similar to the proximity spaces studied by [10]. It has been recently noticed that the theory of proximity spaces is very important to the region-based theory of space. See [15, 6, 7, 13, 14] for details. A set Σ of modal formulas is said to be satisfiable in a Boolean frame $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$, in symbols $\mathcal{F} \text{ sat } \Sigma$, iff there exists a Boolean model $\mathcal{M} = \langle A, 0_A, -_A, \cup_A, I, V \rangle$ based on $\mathcal{F}$ such that all modal formulas in Σ are true in $\mathcal{M}$. We shall say that a set Σ of modal formulas is satisfiable in a class $\mathcal{C}$ of Boolean frames, in symbols $\mathcal{C} \text{ sat } \Sigma$, iff Σ is satisfiable in some Boolean frame in $\mathcal{C}$. A modal formula ϕ is said to be a valid consequence of a set Σ of modal formulas in a Boolean frame $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$, in symbols $\Sigma \models_{\mathcal{F}} \phi$, iff for all Boolean models $\mathcal{M} = \langle A, 0_A, -_A, \cup_A, I, V \rangle$ based on $\mathcal{F}$, if all modal formulas in Σ are true in $\mathcal{M}$ then ϕ is true in $\mathcal{M}$. We shall say that a modal formula ϕ is a valid consequence of a set Σ of modal formulas in a class $\mathcal{C}$ of Boolean frames, in symbols $\Sigma \models_{\mathcal{C}} \phi$, iff ϕ is a valid consequence of Σ in all Boolean frames in $\mathcal{C}$. For all sets Φ of modal formulas, $\mathcal{C}_{\Phi}^B$ will denote the class of all Boolean frames on which Φ is valid.

Proposition 3. *Let Φ , Σ be sets of modal formulas and ϕ be a modal formula such that $\Sigma \models_{\mathcal{C}_{\Phi}^B} \phi$. If $BV(\Sigma)$ is finite then there exists a finite subset Σ' of Σ such that $\Sigma' \models_{\mathcal{C}_{\Phi}^B} \phi$.*

Proof. Assume $BV(\Sigma)$ is finite. Consequently, there exist finitely many logically different modal formulas in $BV(\Sigma)$. Hence, there exists a finite subset Σ' of Σ such that $\Sigma' \models_{\mathcal{C}_{\Phi}^B} \phi$.

Proposition 4. *Let $\mathcal{C}$ be a class of Boolean frames, Σ be a set of modal formulas and ϕ , ψ be modal formulas such that $\Sigma \cup \{\phi\} \models_{\mathcal{C}} \psi$. Then $\Sigma \models_{\mathcal{C}} \phi \rightarrow \psi$.*

Proof. The proposition directly follows from the definition of $\models_{\mathcal{C}}$.

4 Correspondence

4.1 From Kripke Frames to Boolean Frames

Let $\mathcal{F} = \langle S, I \rangle$ be a Kripke frame. The Boolean frame over $\mathcal{F}$ is the structure $B(\mathcal{F}) = \langle A', 0_{A'}, -_{A'}, \cup_{A'}, I' \rangle$ defined as follows:

- $\langle A', 0_{A'}, -_{A'}, \cup_{A'} \rangle$ is the Boolean algebra of all subsets of S ,

- I' is the interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I'(P)$ on A' such that $I'(P) = \{(a_1, \dots, a_n) : \text{there exists } s_1 \text{ in } a_1, \dots, \text{there exists } s_n \text{ in } a_n \text{ such that } (s_1, \dots, s_n) \in I(P)\}$.

Remark that $B(\mathcal{F})$ is a Boolean frame.

Proposition 5. *Let $\mathcal{F} = \langle S, I \rangle$ be a Kripke frame and $B(\mathcal{F}) = \langle A', 0_{A'}, -_{A'}, \cup_{A'}, I' \rangle$ be the Boolean frame over $\mathcal{F}$. Let V be a valuation on $\mathcal{F}$ and V' be the valuation on $B(\mathcal{F})$ such that for all Boolean variables x , $V'(x) = V(x)$. Then for all Boolean terms a , $\widetilde{V}'(a) = \widetilde{V}(a)$ and for all modal formulas ϕ , $\langle B(\mathcal{F}), V' \rangle \Vdash \phi$ iff $\langle \mathcal{F}, V \rangle \Vdash \phi$.*

Proof. See the appendix.

4.2 From Boolean Frames to Kripke Frames

Let $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ be a Boolean frame. The Kripke frame over $\mathcal{F}$ is the structure $K(\mathcal{F}) = \langle S', I' \rangle$ defined as follows:

- S' is the set of all ultrafilters of $\langle A, 0_A, -_A, \cup_A \rangle$,
- I' is the interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I'(P)$ on S' such that $I'(P) = \{(U_1, \dots, U_n) : \text{for all } a_1 \text{ in } U_1, \dots, \text{for all } a_n \text{ in } U_n, (a_1, \dots, a_n) \in I(P)\}$.

Remark that $K(\mathcal{F})$ is a Kripke frame.

Proposition 6. *Let $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ be a Boolean frame and $K(\mathcal{F}) = \langle S', I' \rangle$ be the Kripke frame over $\mathcal{F}$. Let V be a valuation on $\mathcal{F}$ and V' be the valuation on $K(\mathcal{F})$ such that for all Boolean variables x , $V'(x) = \{U : V(x) \in U\}$. Then for all Boolean terms a , $\widetilde{V}'(a) = \{U : \widetilde{V}(a) \in U\}$ and for all modal formulas ϕ , $\langle K(\mathcal{F}), V' \rangle \Vdash \phi$ iff $\langle \mathcal{F}, V \rangle \Vdash \phi$.*

Proof. See the appendix.

4.3 Kripke Frames and Boolean Frames

We now shall consider more closely the ways in which Kripke frames and Boolean frames are alike.

Proposition 7. *Let $\mathcal{F} = \langle S, I \rangle$ be a Kripke frame, $B(\mathcal{F}) = \langle A', 0_{A'}, -_{A'}, \cup_{A'}, I' \rangle$ be the Boolean frame over $\mathcal{F}$ and $K(B(\mathcal{F})) = \langle S'', I'' \rangle$ be the Kripke frame over $B(\mathcal{F})$. Then $\mathcal{F}$ is isomorphic to $K(B(\mathcal{F}))$.*

Proof. Let f be the function taking elements of S to elements of S'' as follows: $f(s) = \{a : s \in a\}$. The reader may easily verify that f is an isomorphism from $\mathcal{F}$ to $K(B(\mathcal{F}))$.

Proposition 8. *Let $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ be a Boolean frame, $K(\mathcal{F}) = \langle S', I' \rangle$ be the Kripke frame over $\mathcal{F}$ and $B(K(\mathcal{F})) = \langle A'', 0_{A''}, -_{A''}, \cup_{A''}, I'' \rangle$ be the Boolean frame over $K(\mathcal{F})$. Then $\mathcal{F}$ is isomorphic to a subframe of $B(K(\mathcal{F}))$.*

Proof. Let f be the function taking elements of A to elements of A'' as follows: $f(a) = \{U : a \in U\}$. The reader may easily verify that f is an injective homomorphism from $\mathcal{F}$ to $B(K(\mathcal{F}))$.

The following is a list of properties of a binary relation symbol P that are interpreted over Kripke frames $\mathcal{F} = \langle S, I \rangle$:

1. For all s in S , $(s, s) \in I(P)$.
2. For all s_1, s_2 in S , if $(s_1, s_2) \in I(P)$ then $(s_2, s_1) \in I(P)$.
3. For all s_1, s_2 in S , if for some s_3 in S , $(s_1, s_3) \in I(P)$ and $(s_3, s_2) \in I(P)$ then $(s_1, s_2) \in I(P)$.
4. There exist s_1, s_2 in S such that $(s_1, s_2) \in I(P)$.
5. For all s_1 in S , there exists s_2 in S such that $(s_1, s_2) \in I(P)$.
6. For all s_2 in S , there exists s_1 in S such that $(s_1, s_2) \in I(P)$.
7. For all s_1, s_2 in S , $(s_1, s_2) \in I(P)$ iff $s_1 = s_2$.
8. For all s_1, s_2 in S , $(s_1, s_2) \in I(P)$.
9. For all s_1, s_2 in S , for some integer $n \geq 0$ and for some $t_0, \dots, t_n$ in S , $t_0 = s_1$, $t_n = s_2$ and for every integer $i \geq 0$, if $1 \leq i \leq n$ then $(t_{i-1}, t_i) \in I(P)$.

The following is a list of properties of a binary relation symbol P that are interpreted over Boolean frames $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$:

1. For all a in A , if $a \neq 0_A$ then $(a, a) \in I(P)$.
2. For all a_1, a_2 in A , if $(a_1, a_2) \in I(P)$ then $(a_2, a_1) \in I(P)$.
3. For all a_1, a_2 in A , if for every a_3 in A , $(a_1, a_3) \in I(P)$ or $(-_A a_3, a_2) \in I(P)$ then $(a_1, a_2) \in I(P)$.
4. $(1_A, 1_A) \in I(P)$.
5. For all a_1 in A , if $a_1 \neq 0_A$ then $(a_1, 1_A) \in I(P)$.
6. For all a_2 in A , if $a_2 \neq 0_A$ then $(1_A, a_2) \in I(P)$.
7. For all a_1, a_2 in A , $(a_1, a_2) \in I(P)$ iff $a_1 \cap_A a_2 \neq 0_A$.
8. For all a_1, a_2 in A , if $a_1 \neq 0_A$ and $a_2 \neq 0_A$ then $(a_1, a_2) \in I(P)$.
9. For all a in A , if $a \neq 0_A$ and $-_A a \neq 0_A$ then $(a, -_A a) \in I(P)$.

Proposition 9. *Let $\mathcal{F} = \langle S, I \rangle$ be a Kripke frame and $B(\mathcal{F}) = \langle A', 0_{A'}, -_{A'}, \cup_{A'}, I' \rangle$ be the Boolean frame over $\mathcal{F}$. Then for all integers $i \geq 0$, if $1 \leq i \leq 9$ then $\mathcal{F}$ satisfies the i -th property iff $B(\mathcal{F})$ satisfies the i -th property.*

Proof. See the appendix.

Proposition 10. *Let $\mathcal{F} = \langle A, 0_A, -_A, \cup_A, I \rangle$ be a Boolean frame and $K(\mathcal{F}) = \langle S', I' \rangle$ be the Kripke frame over $\mathcal{F}$. Then for all integers $i \geq 0$, if $1 \leq i \leq 9$ then $\mathcal{F}$ satisfies the i -th property iff $K(\mathcal{F})$ satisfies the i -th property.*

Proof. See the appendix.

5 Decidability/complexity

5.1 Lower Bound

Let Φ be a set of modal formulas. In this section, we investigate the decidability/complexity of the following decision problem:

- Input: A finite set Σ of modal formulas.
- Output: Determine whether $\mathcal{C}_\Phi^K \text{ sat } \Sigma$.

Proposition 11. *If $\mathcal{C}_\Phi^K$ is nonempty then the above decision problem is NP-hard.*

Proof. Assume $\mathcal{C}_\Phi^K$ is nonempty. The reader may easily verify that for all Boolean terms a , a is a consistent Boolean term of Boolean logic iff $\mathcal{C}_\Phi^K \text{ sat } \{a \neq 0\}$. Since the consistency of Boolean terms of Boolean logic is NP-hard [12], then the above decision problem is NP-hard.

5.2 Filtration

Let Σ be a finite set of modal formulas. Given a Kripke frame $\mathcal{F} = \langle S, I \rangle$ and a valuation V on $\mathcal{F}$, let $\equiv^*$ be the equivalence relation on S defined as follows:

- $s \equiv^* t$ iff for all Boolean variables x in $BV(\Sigma)$, $s \in V(x)$ iff $t \in V(x)$.

By induction on the Boolean term a , the reader may easily verify that if $BV(a) \subseteq BV(\Sigma)$ then for all s, t in S , if $s \equiv^* t$ then $s \in \tilde{V}(a)$ iff $t \in \tilde{V}(a)$. Remark that the function f from the set $\{|s|_{\equiv^*} : s \in S\}$ of all equivalence classes of elements of S modulo $\equiv^*$ to $2^{BV(\Sigma)}$ such that $f(|s|_{\equiv^*}) = \{x : s \in V(x)\}$ is injective. Consequently, $\text{Card}(\{|s|_{\equiv^*} : s \in S\}) \leq 2^{\text{Card}(BV(\Sigma))}$. Let $\mathcal{F}^* = \langle S^*, I^* \rangle$ be the structure defined as follows:

- S^* is the set $\{|s|_{\equiv^*} : s \in S\}$ of all equivalence classes of elements of S modulo $\equiv^*$,
- I^* is the interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I^*(P)$ on S^* such that $I^*(P) = \{(|s_1|_{\equiv^*}, \dots, |s_n|_{\equiv^*}) : \text{there exists } t_1 \text{ in } |s_1|_{\equiv^*}, \dots, \text{there exists } t_n \text{ in } |s_n|_{\equiv^*} \text{ such that } (t_1, \dots, t_n) \in I(P)\}$.

Remark that $\mathcal{F}^*$ is a Kripke frame. Let V^* be the valuation on $\mathcal{F}^*$ defined as follows:

- V^* is the interpretation function mapping the Boolean variables in $BV(\Sigma)$ to subsets of S^* such that $V^*(x) = \{|s|_{\equiv^*} : s \in V(x)\}$.

$\mathcal{F}^*$ and V^* are called the filtration of $\mathcal{F}$ and V through Σ .

Proposition 12. *For all Boolean terms a , if $BV(a) \subseteq BV(\Sigma)$ then $\tilde{V}^*(a) = \{|s|_{\equiv^*} : s \in \tilde{V}(a)\}$ and for all modal formulas ϕ , if $BV(\phi) \subseteq BV(\Sigma)$ then $\langle \mathcal{F}^*, V^* \rangle \Vdash \phi$ iff $\langle \mathcal{F}, V \rangle \Vdash \phi$. Moreover, if $\mathcal{F} \in \mathcal{C}_\Phi^K$ then $\mathcal{F}^* \in \mathcal{C}_\Phi^K$.*

Proof. See the appendix.

5.3 Upper Bound

Proposition 13. *If Φ is finite then the decision problem considered in section 5.1 is in NEXPTIME.*

Proof. Assume Φ is finite. It suffices to prove the existence of an algorithm in NEXPTIME that solves the decision problem considered in section 5.1. Let us consider the following algorithm:

1. Choose a Kripke frame $\mathcal{F} = \langle S, I \rangle$ such that $\text{Card}(S) \leq 2^{\text{Card}(BV(\Sigma))}$.
2. Check whether $\mathcal{F} \in \mathcal{C}_{\Phi}^K$.
3. Check whether Σ is satisfiable in $\mathcal{F}$.

The reader may easily verify that the following decision problem:

- Input: A finite Kripke frame $\mathcal{F} = \langle S, I \rangle$.
- Output: Determine whether $\mathcal{F} \in \mathcal{C}_{\Phi}^K$.

is in *coNP* and the following decision problem:

- Input: A finite Kripke frame $\mathcal{F} = \langle S, I \rangle$ and a finite set Σ of modal formulas.
- Output: Determine whether Σ is satisfiable in $\mathcal{F}$.

is in *NP*. Consequently, the above algorithm can be executed in nondeterministic exponential time.

6 Axiomatization/Completeness

6.1 Axiomatization

To make all the above notions into a formal system, we need axioms and rules of inference. Let Φ be a set of modal formulas. The axioms for $\mathcal{L}_{\Phi}$ are divided into 7 groups:

1. Sentential axioms: Every modal formula which can be obtained from a tautology of propositional classical logic by simultaneously and uniformly substituting modal formulas for the sentence symbols it contains is an axiom for $\mathcal{L}_{\Phi}$.
2. Identity axioms: For all Boolean terms a, a_1, a_2, a_3 , the modal formulas
 - $a \equiv a$,
 - $a_1 \equiv a_2 \rightarrow a_2 \equiv a_1$,
 - $a_1 \equiv a_3 \wedge a_3 \equiv a_2 \rightarrow a_1 \equiv a_2$,
 are axioms for $\mathcal{L}_{\Phi}$.
3. Congruence axioms: For all Boolean terms a, a_1, a_2, b, b_1, b_2 , the modal formulas
 - $a \equiv b \rightarrow -a \equiv -b$,
 - $a_1 \equiv b_1 \wedge a_2 \equiv b_2 \rightarrow a_1 \cup a_2 \equiv b_1 \cup b_2$,
 are axioms for $\mathcal{L}_{\Phi}$.

4. Boolean axioms: For all Boolean terms a, b , if a and b are equivalent Boolean terms of Boolean logic then the modal formula
 - $a \equiv b$,
 is an axiom for $\mathcal{L}_\Phi$.
5. Nondegenerate axiom: The modal formula
 - $0 \neq 1$,
 is an axiom for $\mathcal{L}_\Phi$.
6. Proximity axioms: If $\rho(P) = n$ then for all integers $i \geq 0$, if $1 \leq i \leq n$ then for all Boolean terms $a_1, \dots, a_{i-1}, a_i, a'_i, a''_i, a_{i+1}, \dots, a_n$, the modal formulas
 - $P(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n) \rightarrow a_i \neq 0$,
 - $P(a_1, \dots, a_{i-1}, (a'_i \cup a''_i), a_{i+1}, \dots, a_n) \leftrightarrow (P(a_1, \dots, a_{i-1}, a'_i, a_{i+1}, \dots, a_n) \vee P(a_1, \dots, a_{i-1}, a''_i, a_{i+1}, \dots, a_n))$,
 are axioms for $\mathcal{L}_\Phi$.
7. Φ -axioms: Every modal formula which can be obtained from a modal formula of Φ by simultaneously and uniformly substituting Boolean terms for the Boolean variables it contains is an axiom for $\mathcal{L}_\Phi$.

There is one rule of inference for $\mathcal{L}_\Phi$:

- Modus ponens: From ϕ and $\phi \rightarrow \psi$, infer ψ .

Now, consider a set Σ of modal formulas. A modal formula ϕ is said to be $\mathcal{L}_\Phi$ -deducible from Σ , in symbols $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$, iff there exists a list $\phi_1, \dots, \phi_k$ of modal formulas such that $\phi_k = \phi$ and for all integers $i \geq 0$, if $1 \leq i \leq k$ then either ϕ_i is an axiom for $\mathcal{L}_\Phi$, or ϕ_i belongs to Σ , or ϕ_i is inferred from earlier modal formulas in the list by modus ponens. The list $\phi_1, \dots, \phi_k$ is called a $\mathcal{L}_\Phi$ -deduction of ϕ from Σ . We shall say that Σ is $\mathcal{L}_\Phi$ -consistent iff there exists a modal formula ϕ such that $\Sigma \not\vdash_{\mathcal{L}_\Phi} \phi$. Σ is said to be $\mathcal{L}_\Phi$ -maximal iff Σ is $\mathcal{L}_\Phi$ -consistent and for all $\mathcal{L}_\Phi$ -consistent sets Σ' of modal formulas, if $\Sigma \subseteq \Sigma'$ then $\Sigma = \Sigma'$. We shall say that Φ is coherent iff the set of all $\mathcal{L}_\Phi$ -deducible modal formulas is $\mathcal{L}_\Phi$ -consistent.

Proposition 14. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$. Then there exists a finite subset Σ' of Σ such that $\Sigma' \vdash_{\mathcal{L}_\Phi} \phi$.*

Proof. The proposition directly follows from the definition of $\vdash_{\mathcal{L}_\Phi}$.

Proposition 15. *Let Σ be a set of modal formulas and ϕ, ψ be modal formulas such that $\Sigma \cup \{\phi\} \vdash_{\mathcal{L}_\Phi} \psi$. Then $\Sigma \vdash_{\mathcal{L}_\Phi} \phi \rightarrow \psi$.*

Proof. The proof can be obtained from that given in [9] for the propositional classical logic.

Proposition 16. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$. Then $\Sigma \Vdash_{\mathcal{C}_\Phi^K} \phi$.*

Proof. By induction on the length of a $\mathcal{L}_\Phi$ -deduction of ϕ from Σ , the reader may easily verify that $\Sigma \Vdash_{\mathcal{C}_\Phi^K} \phi$.

Proposition 17. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$. Then $\Sigma \Vdash_{\mathcal{C}_\Phi^B} \phi$.*

Proof. By induction on the length of a $\mathcal{L}_\Phi$ -deduction of ϕ from Σ , the reader may easily verify that $\Sigma \Vdash_{\mathcal{C}_\Phi^B} \phi$.

To end this section, we present some useful results.

Proposition 18. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \not\vdash_{\mathcal{L}_\Phi} \phi$. Then $\Sigma \cup \{\neg\phi\}$ is $\mathcal{L}_\Phi$ -consistent.*

Proof. For the sake of the contradiction, assume $\Sigma \cup \{\neg\phi\}$ is not $\mathcal{L}_\Phi$ -consistent. Consequently, $\Sigma \cup \{\neg\phi\} \vdash_{\mathcal{L}_\Phi} \phi$. By proposition 15, $\Sigma \vdash_{\mathcal{L}_\Phi} \neg\phi \rightarrow \phi$. Hence, $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$: a contradiction.

Proposition 19. *Let Σ be a set of modal formulas such that Σ is $\mathcal{L}_\Phi$ -consistent. Then there exists a $\mathcal{L}_\Phi$ -maximal set Σ' of modal formulas such that $\Sigma \subseteq \Sigma'$.*

Proof. The proof can be obtained from that given in 4 for the propositional classical logic.

6.2 Canonical Model

Assume Φ is coherent. Let Σ be a $\mathcal{L}_\Phi$ -maximal set of modal formulas. The canonical Kripke frame defined by Σ is the structure $\mathcal{F}_\Sigma = \langle S_\Sigma, I_\Sigma \rangle$ defined as follows:

- S_Σ is the set of all maximal sets s of Boolean terms of Boolean logic such that for all Boolean terms a in s , $a \neq 0 \in \Sigma$,
- I_Σ is the interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I_\Sigma(P)$ on S_Σ such that $I_\Sigma(P) = \{(s_1, \dots, s_n) : \text{for all Boolean terms } a_1 \text{ in } s_1, \dots, \text{for all Boolean terms } a_n \text{ in } s_n, P(a_1, \dots, a_n) \in \Sigma\}$.

Remark that $\mathcal{F}_\Sigma$ is a Kripke frame. The canonical valuation defined by Σ is the valuation V_Σ on $\mathcal{F}_\Sigma$ defined as follows:

- V_Σ is the interpretation function mapping the Boolean variables to subsets of S_Σ such that $V_\Sigma(x) = \{s : x \in s\}$.

Proposition 20. *For all Boolean terms a , $\widetilde{V}_\Sigma(a) = \{s : a \in s\}$ and for all modal formulas ϕ , $\langle \mathcal{F}_\Sigma, V_\Sigma \rangle \Vdash \phi$ iff $\phi \in \Sigma$.*

Proof. See the appendix.

6.3 Completeness with Respect to the Kripke Semantics

Assume Φ is coherent.

Proposition 21. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \Vdash_{\mathcal{C}_\Phi^K} \phi$. If $BV(\Sigma)$ is finite then $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$.*

Proof. For the sake of the contradiction, assume $BV(\Sigma)$ is finite and $\Sigma \not\vdash_{\mathcal{L}_\Phi} \phi$. By proposition [18](#), $\Sigma \cup \{\neg\phi\}$ is $\mathcal{L}_\Phi$ -consistent. By proposition [19](#), there exists a $\mathcal{L}_\Phi$ -maximal set Σ' of modal formulas such that $\Sigma \cup \{\neg\phi\} \subseteq \Sigma'$. Remark that for all modal formulas $\psi(x_1, \dots, x_n)$ in Φ and for all Boolean terms $a_1, \dots, a_n$, $\psi(a_1, \dots, a_n) \in \Sigma'$. Let $\mathcal{F}_{\Sigma'} = \langle S_{\Sigma'}, I_{\Sigma'} \rangle$ be the canonical Kripke frame defined by Σ' and $V_{\Sigma'}$ be the canonical valuation defined by Σ' . By proposition [20](#), for all Boolean terms a , $\widetilde{V_{\Sigma'}}(a) = \{s : a \in s\}$ and for all modal formulas ψ , $\langle \mathcal{F}_{\Sigma'}, V_{\Sigma'} \rangle \models \psi$ iff $\psi \in \Sigma'$. Let $\mathcal{F}_{\Sigma'}^*$ and $V_{\Sigma'}^*$ be the filtration of $\mathcal{F}_{\Sigma'}$ and $V_{\Sigma'}$ through $\Sigma \cup \{\neg\phi\}$. By proposition [12](#), for all modal formulas ψ in $\Sigma \cup \{\neg\phi\}$, $\langle \mathcal{F}_{\Sigma'}^*, V_{\Sigma'}^* \rangle \models \psi$. Consequently, to prove the proposition, it suffices to demonstrate that $\mathcal{F}_{\Sigma'}^* \in \mathcal{C}_\Phi^K$. For the sake of the contradiction, assume $\mathcal{F}_{\Sigma'}^* \notin \mathcal{C}_\Phi^K$. Hence, Φ is not valid on $\mathcal{F}_{\Sigma'}^*$. By proposition [12](#), for all modal formulas $\psi(x_1, \dots, x_n)$ in Φ and for all Boolean terms $a_1, \dots, a_n$, if $BV(a_1) \subseteq BV(\Sigma \cup \{\neg\phi\})$, $\dots$, $BV(a_n) \subseteq BV(\Sigma \cup \{\neg\phi\})$ then $\langle \mathcal{F}_{\Sigma'}^*, V_{\Sigma'}^* \rangle \models \psi(a_1, \dots, a_n)$. Non validity of Φ on $\mathcal{F}_{\Sigma'}^*$ implies that there exists a modal formula $\psi(x_1, \dots, x_n)$ in Φ and there exists a valuation V' on $\mathcal{F}_{\Sigma'}^*$ such that $\langle \mathcal{F}_{\Sigma'}^*, V' \rangle \not\models \psi(x_1, \dots, x_n)$. For all integers $i \geq 0$, if $1 \leq i \leq n$ then let $a_i = \bigcup \{b(s) : s \in V'(x_i)\}$ where $b(s) = \bigcap \{x : x \in BV(\Sigma \cup \{\neg\phi\}) \text{ and } s \in V'(x) \cap \bigcap \{-x : x \in BV(\Sigma \cup \{\neg\phi\}) \text{ and } s \notin V'(x)\}\}$. The reader may easily verify that $BV(a_1) \subseteq BV(\Sigma \cup \{\neg\phi\})$, $\dots$, $BV(a_n) \subseteq BV(\Sigma \cup \{\neg\phi\})$. Therefore, $\langle \mathcal{F}_{\Sigma'}^*, V_{\Sigma'}^* \rangle \models \psi(a_1, \dots, a_n)$. Remark that $\widetilde{V_{\Sigma'}^*}(a_1) = V'(x_1)$, $\dots$, $\widetilde{V_{\Sigma'}^*}(a_n) = V'(x_n)$. Thus, $\langle \mathcal{F}_{\Sigma'}^*, V' \rangle \models \psi(x_1, \dots, x_n)$: a contradiction.

6.4 Completeness with Respect to the Boolean Semantics

Assume Φ is coherent.

Proposition 22. *Let Σ be a set of modal formulas and ϕ be a modal formula such that $\Sigma \vdash_{\mathcal{C}_\Phi^B} \phi$. If $BV(\Sigma)$ is finite then $\Sigma \vdash_{\mathcal{L}_\Phi} \phi$.*

Proof. For the sake of the contradiction, assume $\Sigma \not\vdash_{\mathcal{L}_\Phi} \phi$. By proposition [18](#), $\Sigma \cup \{\neg\phi\}$ is $\mathcal{L}_\Phi$ -consistent. By proposition [19](#), there exists a $\mathcal{L}_\Phi$ -maximal set Σ' of modal formulas such that $\Sigma \cup \{\neg\phi\} \subseteq \Sigma'$. Remark that for all modal formulas $\psi(x_1, \dots, x_n)$ in Φ and for all Boolean terms $a_1, \dots, a_n$, $\psi(a_1, \dots, a_n) \in \Sigma'$. We define the equivalence relation $\equiv_{\Sigma'}$ on the set of all Boolean terms in $BV(\Sigma \cup \{\neg\phi\})$ as follows:

- $a_1 \equiv_{\Sigma'} a_2$ iff $a_1 \equiv a_2 \in \Sigma'$.

Let $\mathcal{F}_{\Sigma'} = \langle A_{\Sigma'}, 0_{A_{\Sigma'}}, -_{A_{\Sigma'}}, \cup_{A_{\Sigma'}}, I_{\Sigma'} \rangle$ be the structure defined as follows:

- $\langle A_{\Sigma'}, 0_{A_{\Sigma'}}, -_{A_{\Sigma'}}, \cup_{A_{\Sigma'}} \rangle$ is the Boolean algebra of all equivalence classes of Boolean terms in $BV(\Sigma \cup \{\neg\phi\})$ modulo $\equiv_{\Sigma'}$,
- $I_{\Sigma'}$ is the interpretation function mapping the relation symbols P of $\mathcal{R}$ to appropriate relations $I_{\Sigma'}(P)$ on $A_{\Sigma'}$ such that $I_{\Sigma'}(P) = \{(| a_1 |_{\equiv_{\Sigma'}}, \dots, | a_n |_{\equiv_{\Sigma'}}) : P(a_1, \dots, a_n) \in \Sigma'\}$.

Remark that $\mathcal{F}_{\Sigma'}$ is a Boolean frame. Let $V_{\Sigma'}$ be the valuation on $\mathcal{F}_{\Sigma'}$ defined as follows:

- $V_{\Sigma'}$ is the interpretation function mapping the Boolean variables in $BV(\Sigma \cup \{\neg\phi\})$ to elements of $A_{\Sigma'}$ such that $V_{\Sigma'}(x) = |x|_{\equiv_{\Sigma'}}$.

By induction on the Boolean term a in $BV(\Sigma \cup \{\neg\phi\})$, the reader may easily verify that $\widetilde{V_{\Sigma'}}(a) = |a|_{\equiv_{\Sigma'}}$ and by induction on the modal formula ψ in $BV(\Sigma \cup \{\neg\phi\})$, the reader may easily verify that $\langle \mathcal{F}_{\Sigma'}, V_{\Sigma'} \rangle \models \psi$ iff $\psi \in \Sigma'$. Consequently, for all modal formulas ψ in $\Sigma \cup \{\neg\phi\}$, $\langle \mathcal{F}_{\Sigma'}, V_{\Sigma'} \rangle \models \psi$. Hence, to prove the proposition, it suffices to demonstrate that $\mathcal{F}_{\Sigma'} \in \mathcal{C}_{\Phi}^B$. For the sake of the contradiction, assume $\mathcal{F}_{\Sigma'} \notin \mathcal{C}_{\Phi}^B$. Hence, Φ is not valid on $\mathcal{F}_{\Sigma'}$. Remark that for all modal formulas $\psi(x_1, \dots, x_n)$ in Φ and for all Boolean terms $a_1, \dots, a_n$, if $BV(a_1) \subseteq BV(\Sigma \cup \{\neg\phi\})$, $\dots$, $BV(a_n) \subseteq BV(\Sigma \cup \{\neg\phi\})$ then $\langle \mathcal{F}_{\Sigma'}, V_{\Sigma'} \rangle \models \psi(a_1, \dots, a_n)$. Non validity of Φ on $\mathcal{F}_{\Sigma'}$ implies that there exists a modal formula $\psi(x_1, \dots, x_n)$ in Φ and there exists a valuation V' on $\mathcal{F}_{\Sigma'}$ such that $\langle \mathcal{F}_{\Sigma'}, V' \rangle \not\models \psi(x_1, \dots, x_n)$. For all integers $i \geq 0$, if $1 \leq i \leq n$ then let $a_i = \bigcup \{b(s) : s \in V'(x_i)\}$ where $b(s) = \bigcap \{x : x \in BV(\Sigma \cup \{\neg\phi\}) \text{ and } s \in V'(x)\} \cap \bigcap \{-x : x \in BV(\Sigma \cup \{\neg\phi\}) \text{ and } s \notin V'(x)\}$. The reader may easily verify that $BV(a_1) \subseteq BV(\Sigma \cup \{\neg\phi\})$, $\dots$, $BV(a_n) \subseteq BV(\Sigma \cup \{\neg\phi\})$. Therefore, $\langle \mathcal{F}_{\Sigma'}, V_{\Sigma'} \rangle \models \psi(a_1, \dots, a_n)$. Remark that $\widetilde{V_{\Sigma'}}(a_1) = V'(x_1)$, $\dots$, $\widetilde{V_{\Sigma'}}(a_n) = V'(x_n)$. Thus, $\langle \mathcal{F}_{\Sigma'}, V' \rangle \models \psi(x_1, \dots, x_n)$: a contradiction.

7 Canonicity

Let Φ be a coherent set of modal formulas. We shall say that the formal system $\mathcal{L}_{\Phi}$ is weakly canonical iff there exists a $\mathcal{L}_{\Phi}$ -maximal set Σ of modal formulas such that the canonical Kripke frame $\mathcal{F}_{\Sigma} = \langle S_{\Sigma}, I_{\Sigma} \rangle$ defined by Σ is in $\mathcal{C}_{\Phi}^K$. $\mathcal{L}_{\Phi}$ is said to be strongly canonical iff for all $\mathcal{L}_{\Phi}$ -maximal sets Σ of modal formulas, the canonical Kripke frame $\mathcal{F}_{\Sigma} = \langle S_{\Sigma}, I_{\Sigma} \rangle$ defined by Σ is in $\mathcal{C}_{\Phi}^K$.

Proposition 23. *Let P be a binary relation symbol. If Φ is a subset of the set of modal formulas containing the following modal formulas:*

- $x \neq 0 \rightarrow P(x, x)$,
- $P(x, y) \rightarrow P(y, x)$,
- $P(1, 1)$,
- $x \neq 0 \rightarrow P(x, 1)$,
- $y \neq 0 \rightarrow P(1, y)$,
- $P(x, y) \leftrightarrow x \cap y \neq 0$,
- $x \neq 0 \wedge y \neq 0 \rightarrow P(x, y)$,

then $\mathcal{L}_{\Phi}$ is strongly canonical.

Proof. We illustrate with the case of the set $\{P(1, 1)\}$.

For the sake of the contradiction, assume $\mathcal{L}_{\{P(1, 1)\}}$ is not strongly canonical. Consequently, there exists a $\mathcal{L}_{\{P(1, 1)\}}$ -maximal set Σ of modal formulas such that

the canonical Kripke frame $\mathcal{F}_\Sigma = \langle S_\Sigma, I_\Sigma \rangle$ defined by Σ is not in $\mathcal{C}_{\{P(1,1)\}}^K$. By proposition 20, $\langle \mathcal{F}_\Sigma, V_\Sigma \rangle \Vdash P(1, 1)$. Hence, for all valuations V on $\mathcal{F}_\Sigma$, $\langle \mathcal{F}_\Sigma, V \rangle \Vdash P(1, 1)$. Therefore, $\mathcal{F}_\Sigma \in \mathcal{C}_{\{P(1,1)\}}^K$: a contradiction.

Proposition 24. *Let P be a binary relation symbol. If Φ is the set of modal formulas containing the following modal formulas:*

- $x \neq 0 \rightarrow P(x, x)$,
- $P(x, y) \rightarrow P(y, x)$,
- $x \neq 0 \wedge -x \neq 0 \rightarrow P(x, -x)$,

then $\mathcal{L}_\Phi$ is weakly canonical and not strongly canonical.

Proof. The reader may easily verify that for all Kripke frames $\mathcal{F} = \langle S, I \rangle$, $\mathcal{F} \Vdash \Phi$ iff $\mathcal{F}$ satisfies the following properties:

- For all s in S , $(s, s) \in I(P)$,
- For all s_1, s_2 in S , if $(s_1, s_2) \in I(P)$ then $(s_2, s_1) \in I(P)$,
- For all s_1, s_2 in S , for some integer $n \geq 0$ and for some $t_0, \dots, t_n$ in S , $t_0 = s_1$, $t_n = s_2$ and for every integer $i \geq 0$, if $1 \leq i \leq n$ then $(t_{i-1}, t_i) \in I(P)$.

Let $x_1, x_2, \dots$, be a list of the set of all Boolean variables. If s is a maximal set of Boolean terms of Boolean logic then we use $(s^i)_{1 \leq i}$ to denote the list of Boolean literals defined as follows:

- For all integers $i \geq 0$, if $1 \leq i$ then if $x_i \in s$ then $s^i = x_i$ else $s^i = -x_i$.

The reader may easily verify that for all $\mathcal{L}_\Phi$ -maximal sets Σ of modal formulas, the canonical Kripke frame $\mathcal{F}_\Sigma = \langle S_\Sigma, I_\Sigma \rangle$ defined by Σ is such that S_Σ is the set of all maximal sets s of Boolean terms of Boolean logic such that for all integers $i \geq 0$, if $1 \leq i$ then $s^1 \cap \dots \cap s^i \neq 0 \in \Sigma$ and I_Σ is the interpretation function mapping the binary relation symbol P to the appropriate binary relation $I_\Sigma(P)$ on S_Σ such that $I_\Sigma(P) = \{(s_1, s_2): \text{for all integers } i \geq 0, \text{ if } 1 \leq i \text{ then } P(s_1^1 \cap \dots \cap s_1^i, s_2^1 \cap \dots \cap s_2^i) \in \Sigma\}$. For all maximal sets s_1, s_2 of Boolean terms of Boolean logic and for all integers $i \geq 0$, if $1 \leq i$ then let $\text{dist}^i(s_1, s_2)$ be the number of integers $j \geq 0$ such that $1 \leq j \leq i$ and $s_1^j \neq s_2^j$.

Let $\Sigma_1 = \{s^1 \cap \dots \cap s^i \neq 0: s \text{ is a maximal set of Boolean terms of Boolean logic and } i \geq 0 \text{ is an integer such that } 1 \leq i\} \cup \{P(s_1^1 \cap \dots \cap s_1^i, s_2^1 \cap \dots \cap s_2^i): s_1 \text{ and } s_2 \text{ are maximal sets of Boolean terms of Boolean logic and } i \geq 0 \text{ is an integer such that } 1 \leq i\}$. The reader may easily verify that Σ_1 is $\mathcal{L}_\Phi$ -consistent. By proposition 19, there exists a $\mathcal{L}_\Phi$ -maximal set Σ'_1 of modal formulas such that $\Sigma_1 \subseteq \Sigma'_1$. The reader may easily verify that the canonical Kripke frame $\mathcal{F}_{\Sigma'_1} = \langle S_{\Sigma'_1}, I_{\Sigma'_1} \rangle$ defined by Σ'_1 is in $\mathcal{C}_\Phi^K$.

Let $\Sigma_2 = \{s^1 \cap \dots \cap s^i \neq 0: s \text{ is a maximal set of Boolean terms of Boolean logic and } i \geq 0 \text{ is an integer such that } 1 \leq i\} \cup \{P(s_1^1 \cap \dots \cap s_1^i, s_2^1 \cap \dots \cap s_2^i): s_1 \text{ and } s_2 \text{ are maximal sets of Boolean terms of Boolean logic and } i \geq 0 \text{ is an integer such that } 1 \leq i \text{ and } \text{dist}^i(s_1, s_2) \leq 1\} \cup \{\neg P(s_1^1 \cap \dots \cap s_1^i, s_2^1 \cap \dots \cap s_2^i): s_1 \text{ and } s_2 \text{ are maximal sets of Boolean terms of Boolean logic and } i \geq 0 \text{ is an integer}\}$.

such that $1 \leq i$ and $\text{dist}^i(s_1, s_2) \geq 2\}$. The reader may easily verify that Σ_2 is $\mathcal{L}_\Phi$ -consistent. By proposition [19], there exists a $\mathcal{L}_\Phi$ -maximal set Σ'_2 of modal formulas such that $\Sigma_2 \subseteq \Sigma'_2$. The reader may easily verify that the canonical Kripke frame $\mathcal{F}_{\Sigma'_2} = \langle S_{\Sigma'_2}, I_{\Sigma'_2} \rangle$ defined by Σ'_2 is not in $\mathcal{C}_\Phi^K$.

8 Variants and Open Problems

Concerning decidability and complexity, we have proved in section [5] that if Φ is finite then the satisfiability problem with respect to $\mathcal{L}_\Phi$ is *NP*-hard and in *NEXPTIME*. In [2], we have proved that there exist sets Φ of modal formulas such that the satisfiability problem with respect to $\mathcal{L}_\Phi$ is *NP*-complete and there exist sets Φ of modal formulas such that the satisfiability problem with respect to $\mathcal{L}_\Phi$ is *PSPACE*-complete. Does there exist a set Φ of modal formulas such that the satisfiability problem with respect to $\mathcal{L}_\Phi$ is *EXPTIME*-complete or *NEXPTIME*-complete?

Concerning axiomatization and completeness, we have proved in section [6] that if Φ is coherent then the axioms and rules considered in section [6.1] constitute a complete formal system $\mathcal{L}_\Phi$. We conjecture that given a finite set Φ of modal formulas, it is decidable in nondeterministic polynomial time to determine whether Φ is coherent.

Concerning canonicity, we have proved in section [7] that there exist weakly canonical and strongly canonical formal systems $\mathcal{L}_\Phi$ and there exist weakly canonical and not strongly canonical formal systems $\mathcal{L}_\Phi$. We conjecture that all formal systems $\mathcal{L}_\Phi$ are weakly canonical.

References

1. Balbiani, P., Tinchev, T., Vakarelov, D.: Dynamic logics of the region-based theory of discrete spaces. *Journal of Applied Non-Classical Logics* 17 (2007)
2. Balbiani, P., Tinchev, T., Vakarelov, D.: Modal logics for region-based theories of space. *Fundamenta Informaticæ* 81 (2007)
3. Chagrov, A., Rybakov, M.: How many variables does one need to prove *PSPACE*-hardness of modal logics? In: Balbiani, P., Suzuki, N.-Y., Wolter, F., Zakharyashev, M. (eds.) *Advances in Modal Logic*, vol. 4, King's College (2003)
4. Chang, C., Keisler, H.: *Model Theory*. Elsevier, Amsterdam (1990)
5. Dimov, G., Vakarelov, D.: Contact algebras and region-based theory of space: a proximity approach – I. *Fundamenta Informaticæ* 74 (2006)
6. Dimov, G., Vakarelov, D.: Contact algebras and region-based theory of space: proximity approach – II. *Fundamenta Informaticæ* 74 (2006)
7. Düntsch, I., Winter, M.: A representation theorem for Boolean contact algebras. *Theoretical Computer Science* 347 (2005)
8. Halpern, J.: The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence* 75 (1995)
9. Kleene, S.: *Introduction to Metamathematics*. North-Holland, Amsterdam (1971)
10. Naimpally, S., Warrack, B.: *Proximity Spaces*. Cambridge University Press, Cambridge (1970)

11. Nguyen, L.: On the complexity of fragments of modal logics. In: Schmidt, R., Pratt-Hartmann, I., Reynolds, M., Wansing, H. (eds.) *Advances in Modal Logic*, vol. 5, King's College (2005)
12. Papadimitriou, C.: *Computational Complexity*. Addison-Wesley, Reading (1994)
13. Stell, J.: Boolean connection algebras: a new approach to the region connection calculus. *Artificial Intelligence* 122 (2000)
14. Vakarelov, D., Dimov, G., Düntsch, I., Bennett, B.: A proximity approach to some region-based theory of space. *Journal of Applied Non-Classical Logics* 12 (2002)

Appendix

In this appendix, we provide the proofs of propositions [5](#), [6](#), [9](#), [10](#), [12](#) and [20](#).

Proof of proposition [5](#). By induction on a , the reader may easily verify that $\widetilde{V}'(a) = \widetilde{V}(a)$ and by induction on ϕ , the reader may easily verify that $\langle B(\mathcal{F}), V' \rangle \Vdash \phi$ iff $\langle \mathcal{F}, V \rangle \Vdash \phi$.

Proof of proposition [6](#). By induction on a , the reader may easily verify that $\widetilde{V}'(a) = \widetilde{V}(a)$. By induction on ϕ , let us verify that $\langle K(\mathcal{F}), V' \rangle \Vdash \phi$ iff $\langle \mathcal{F}, V \rangle \Vdash \phi$. We only consider the base case $P(a_1, \dots, a_n)$.

Assume $\langle K(\mathcal{F}), V' \rangle \Vdash P(a_1, \dots, a_n)$. The reader may easily verify that $\langle \mathcal{F}, V \rangle \Vdash P(a_1, \dots, a_n)$.

Assume $\langle \mathcal{F}, V \rangle \Vdash P(a_1, \dots, a_n)$. Consequently, $(\widetilde{V}(a_1), \dots, \widetilde{V}(a_n)) \in I(P)$. Let $U'_1 = \{b_1: \widetilde{V}(a_1) \leq_A b_1\}$, ..., $U'_n = \{b_n: \widetilde{V}(a_n) \leq_A b_n\}$. The reader may easily verify that $U'_1, \dots, U'_n$ are proper filters of $\langle A, 0_A, -_A, \cup_A \rangle$ such that $\widetilde{V}(a_1) \in U'_1, \dots, \widetilde{V}(a_n) \in U'_n$ and for all b_1 in $U'_1, \dots$, for all b_n in U'_n , $(b_1, \dots, b_n) \in I(P)$. By Zorn's lemma, the reader may define ultrafilters $U_1, \dots, U_n$ of $\langle A, 0_A, -_A, \cup_A \rangle$ such that $\widetilde{V}(a_1) \in U_1, \dots, \widetilde{V}(a_n) \in U_n$ and for all b_1 in $U_1, \dots$, for all b_n in U_n , $(b_1, \dots, b_n) \in I(P)$. Hence, $U_1 \in \widetilde{V}'(a_1), \dots, U_n \in \widetilde{V}'(a_n)$ and $(U_1, \dots, U_n) \in I'(P)$. Therefore, $\langle K(\mathcal{F}), V' \rangle \Vdash P(a_1, \dots, a_n)$.

Proof of proposition [9](#). We illustrate with the case of the 3-rd property.

Assume $\mathcal{F}$ satisfies the 3-rd property. Consequently, for all s_1, s_2 in S , if for some s_3 in S , $(s_1, s_3) \in I(P)$ and $(s_3, s_2) \in I(P)$ then $(s_1, s_2) \in I(P)$. For the sake of the contradiction, assume $B(\mathcal{F})$ does not satisfy the 3-rd property. Hence, there exist a_1, a_2 in A' such that for every a_3 in A' , $(a_1, a_3) \in I'(P)$ or $(-_{A'}a_3, a_2) \in I'(P)$ and $(a_1, a_2) \notin I'(P)$. Let $a = \{s: \text{for all } s_1 \text{ in } a_1, (s_1, s) \notin I(P)\}$. The reader may easily verify that $(a_1, a) \notin I'(P)$. Therefore, $(-_{A'}a, a_2) \in I'(P)$. Thus, there exists s in $-_{A'}a$ and there exists s_2 in a_2 such that $(s, s_2) \in I(P)$. Consequently, there exists s_1 in a_1 such that $(s_1, s) \in I(P)$. Hence, $(s_1, s_2) \in I(P)$. Therefore, $(a_1, a_2) \in I'(P)$: a contradiction.

Assume $B(\mathcal{F})$ satisfies the 3-rd property. Consequently, for all a_1, a_2 in A' , if for every a_3 in A' , $(a_1, a_3) \in I'(P)$ or $(-_{A'}a_3, a_2) \in I'(P)$ then $(a_1, a_2) \in I'(P)$. For the sake of the contradiction, assume $\mathcal{F}$ does not satisfy the 3-rd property. Hence, there exist s_1, s_2 in S such that for some s_3 in S , $(s_1, s_3) \in I(P)$ and

$(s_3, s_2) \in I(P)$ and $(s_1, s_2) \notin I(P)$. Let $a_1 = \{s_1\}$ and $a_2 = \{s_2\}$. The reader may easily verify that for every a in A' , $(a_1, a) \in I'(P)$ or $(-_{A'}a, a_2) \in I'(P)$. Therefore, $(a_1, a_2) \in I'(P)$. Thus, $(s_1, s_2) \in I(P)$: a contradiction.

Proof of proposition 10. We illustrate with the case of the 3-rd property.

Assume $\mathcal{F}$ satisfies the 3-rd property. Consequently, for all a_1, a_2 in A , if for every a_3 in A , $(a_1, a_3) \in I(P)$ or $(-_{A}a_3, a_2) \in I(P)$ then $(a_1, a_2) \in I(P)$. For the sake of the contradiction, assume $K(\mathcal{F})$ does not satisfy the 3-rd property. Hence, there exist U_1, U_2 in S' such that for some U_3 in S' , $(U_1, U_3) \in I'(P)$ and $(U_3, U_2) \in I'(P)$ and $(U_1, U_2) \notin I'(P)$. The reader may easily verify that there exists a_1 in U_1 and there exists a_2 in U_2 such that $(a_1, a_2) \notin I(P)$. Therefore, for some a in A , $(a_1, a) \notin I(P)$ and $(-_{A}a, a_2) \notin I(P)$. Now, we have to consider two cases: $a \in U_3$ or $-_{A}a \in U_3$. In the former case, $(U_1, U_3) \notin I'(P)$: a contradiction. In the latter case, $(U_3, U_2) \notin I'(P)$: a contradiction.

Assume $K(\mathcal{F})$ satisfies the 3-rd property. Consequently, for all U_1, U_2 in S' , if for some U_3 in S' , $(U_1, U_3) \in I'(P)$ and $(U_3, U_2) \in I'(P)$ then $(U_1, U_2) \in I'(P)$. For the sake of the contradiction, assume $\mathcal{F}$ does not satisfy the 3-rd property. Hence, there exist a_1, a_2 in A such that for every a_3 in A , $(a_1, a_3) \in I(P)$ or $(-_{A}a_3, a_2) \in I(P)$ and $(a_1, a_2) \notin I(P)$. Let $U' = \{b: \text{there exist } b', b'' \text{ in } A \text{ such that } (a_1, b') \notin I(P), (-_{A}b'', a_2) \notin I(P) \text{ and } b = -_{A}b' \cap_{A} b''\}$. The reader may easily verify that U' is a proper filter of $\langle A, 0_A, -_{A}, \cup_A \rangle$ such that for every b in U' , $(a_1, b) \in I(P)$ and $(b, a_2) \in I(P)$. By Zorn's lemma, the reader may define an ultrafilter U of $\langle A, 0_A, -_{A}, \cup_A \rangle$ such that for every b in U , $(a_1, b) \in I(P)$ and $(b, a_2) \in I(P)$. Let $U'_1 = \{b_1: a_1 \leq_A b_1\}$ and $U'_2 = \{b_2: a_2 \leq_A b_2\}$. The reader may easily verify that U'_1 and U'_2 are proper filters of $\langle A, 0_A, -_{A}, \cup_A \rangle$ such that $a_1 \in U'_1$, $a_2 \in U'_2$ and for all b_1 in U'_1 and for all b_2 in U'_2 , for every b in U , $(b_1, b) \in I(P)$ and $(b, b_2) \in I(P)$. By Zorn's lemma, the reader may define ultrafilters U_1 and U_2 of $\langle A, 0_A, -_{A}, \cup_A \rangle$ such that $a_1 \in U_1$, $a_2 \in U_2$ and for all b_1 in U_1 and for all b_2 in U_2 , for every b in U , $(b_1, b) \in I(P)$ and $(b, b_2) \in I(P)$. Therefore, $(U_1, U) \in I'(P)$ and $(U, U_2) \in I'(P)$. Thus, $(U_1, U_2) \in I'(P)$. Consequently, $(a_1, a_2) \in I(P)$: a contradiction.

Proof of proposition 12. By induction on the Boolean term a , the reader may easily verify that if $BV(a) \subseteq BV(\Sigma)$ then $\widetilde{V}^*(a) = \{s \mid s \models a\}$ and by induction on the modal formula ϕ , the reader may easily verify that if $BV(\phi) \subseteq BV(\Sigma)$ then $\langle \mathcal{F}^*, V^* \rangle \models \phi$ iff $\langle \mathcal{F}, V \rangle \models \phi$. Hence, to prove the proposition, it suffices to demonstrate that if $\mathcal{F} \in \mathcal{C}_{\Phi}^K$ then $\mathcal{F}^* \in \mathcal{C}_{\Phi}^K$. For the sake of the contradiction, assume $\mathcal{F} \in \mathcal{C}_{\Phi}^K$ and $\mathcal{F}^* \notin \mathcal{C}_{\Phi}^K$. Therefore, Φ is valid on $\mathcal{F}$ and Φ is not valid on $\mathcal{F}^*$. Validity of Φ on $\mathcal{F}$ implies that for all modal formulas $\phi(x_1, \dots, x_n)$ in Φ and for all Boolean terms $a_1, \dots, a_n$, if $BV(a_1) \subseteq BV(\Sigma)$, $\dots$, $BV(a_n) \subseteq BV(\Sigma)$ then $\langle \mathcal{F}^*, V^* \rangle \models \phi(a_1, \dots, a_n)$. Non validity of Φ on $\mathcal{F}^*$ implies that there exists a modal formula $\phi(x_1, \dots, x_n)$ in Φ and there exists a valuation V' on $\mathcal{F}^*$ such that $\langle \mathcal{F}^*, V' \rangle \not\models \phi(x_1, \dots, x_n)$. For all integers $i \geq 0$, if $1 \leq i \leq n$ then let $a_i = \bigcup \{b(s): s \in V'(x_i)\}$ where $b(s) = \bigcap \{x: x \in BV(\Sigma) \text{ and } s \in V'(x)\} \cap \bigcap \{-x: x \in BV(\Sigma) \text{ and } s \notin V'(x)\}$. The

reader may easily verify that $BV(a_1) \subseteq BV(\Sigma)$, ..., $BV(a_n) \subseteq BV(\Sigma)$. Thus, $\langle \mathcal{F}^*, V^* \rangle \Vdash \phi(a_1, \dots, a_n)$. Remark that $\widetilde{V}^*(a_1) = V'(x_1)$, ..., $\widetilde{V}^*(a_n) = V'(x_n)$. Consequently, $\langle \mathcal{F}^*, V' \rangle \Vdash \phi(x_1, \dots, x_n)$: a contradiction.

Proof of proposition 20. By induction on the Boolean term a , the reader may easily verify that $\widetilde{V}_\Sigma(a) = \{s : a \in s\}$. By induction on the modal formula ϕ , let us verify that $\langle \mathcal{F}_\Sigma, V_\Sigma \rangle \Vdash \phi$ iff $\phi \in \Sigma$. We only consider the base case $P(a_1, \dots, a_n)$.

Assume $\langle \mathcal{F}_\Sigma, V_\Sigma \rangle \Vdash P(a_1, \dots, a_n)$. The reader may easily verify that $P(a_1, \dots, a_n) \in \Sigma$.

Assume $P(a_1, \dots, a_n) \in \Sigma$. Let $s'_1 = \{a_1\}$, ..., $s'_n = \{a_n\}$. The reader may easily verify that $s'_1, \dots, s'_n$ are consistent sets of Boolean terms of Boolean logic such that $a_1 \in s'_1$, ..., $a_n \in s'_n$ and for all Boolean terms b_1 in s'_1 , ..., for all Boolean terms b_n in s'_n , $P(b_1, \dots, b_n) \in \Sigma$. By Zorn's lemma, the reader may define maximal sets $s_1, \dots, s_n$ of Boolean terms of Boolean logic such that $a_1 \in s_1$, ..., $a_n \in s_n$ and for all Boolean terms b_1 in s_1 , ..., for all Boolean terms b_n in s_n , $P(b_1, \dots, b_n) \in \Sigma$. Consequently, $s_1 \in \widetilde{V}_\Sigma(a_1)$, ..., $s_n \in \widetilde{V}_\Sigma(a_n)$ and $(s_1, \dots, s_n) \in I_\Sigma(P)$. Hence, $\langle \mathcal{F}_\Sigma, V_\Sigma \rangle \Vdash P(a_1, \dots, a_n)$.

Relation Algebra and RELVIEW in Practical Use: Construction of Special University Timetables

Rudolf Berghammer and Britta Kehden

Institut für Informatik, Christian-Albrechts-Universität Kiel
Olshausenstraße 40, 24098 Kiel, Germany
{rub | bk}@informatik.uni-kiel.de

Abstract. In this paper, we are concerned with a special timetabling problem. It was posed to us by the administration of our university and stems from the adoption of the British-American system of university education in Germany. This change led to the concrete task of constructing a timetable that enables the undergraduate education of secondary school teachers within three years in the “normal case” and within four years in the case of exceptional combinations of fields of study. We develop a relational model of the special timetabling problem and apply the RELVIEW tool to compute solutions.

1 Introduction

The construction of timetables for educational institutions and other purposes is a rich area of research since many years. It has strong links to graph theory, particularly with regard to graph-colouring, network flows, and matching in bipartite graphs. Primarily graph-colouring methods are used as a basis of a lot of timetabling algorithms. See e.g., [4], Sect. 5.6, for an overview.

Concrete timetabling problems frequently are very complex. They also vary widely in their structure. Therefore, people developed abstract specifications that are general enough to cover most concrete cases. Such a specification is e.g., presented in [78]. Unlike most of the abstract timetable specifications it bases on relation algebra in the sense of [10,9] instead of graphs. Given a relation A that specifies whether a meeting can take place in a time slot and a relation P that specifies whether a participant takes part in a meeting, a solution of the timetabling problem for input A and P is a relation S between meetings and time slots that is univalent and total (i.e., a function from meetings to time slots) and fulfils $S \subseteq A$ and $(PP^T \cap \bar{I})S \subseteq \bar{S}$. The first inclusion says that if S assigns a meeting m to time slot h , then m can take place in h , and the second inclusion ensures that if a participant attends two different meetings m and m' (i.e., these are in conflict), then m and m' are assigned to different time slots.

In [5] this relation-algebraic specification of a solution of a timetabling problem is reformulated in such a way that instead of the input relation A between meetings and time slots and the result relation S of the same type their corresponding vectors on the direct product of meetings and time slots are used. Interpreting relations column-wisely as lists of vectors, this approach allowed

the combination of relation algebra and randomized search heuristics and led to relational algorithms, e.g., expressible in the programming language of the RELVIEW tool (see [113]), which can be used for the construction of timetables.

In this paper, we are concerned with the solution of another abstract timetabling problem. It was posed to us by the administration of our university and stems from Germany's agreement to the so-called Bologna accord. A consequence of this accord is the current change from the classical German university education system (normally ending with Diplom or Magister degrees) to the British-American undergraduate-graduate system with Bachelor and Master degrees. Particularly with regard to the undergraduate education of secondary school teachers this change causes some difficulties. One of them is to enable a three years duration of study without to abolish Germany's tradition of (at least) two different fields of study, and exactly this led to the timetabling problem.

Given an informal description, its input data, and some additional desirable properties of possible solutions, we have been asked by the university administration to develop an algorithmic solution of the problem and to test the approach with its help by constructing a timetable that enables a three years duration of undergraduate-study in the case of the most selected combinations of subjects and a four years duration of study in the case of exceptional combinations of subjects. To solve this task, we have developed a relation-algebraic model of the problem. Using ideas of [5], we then have been able to apply the RELVIEW tool for testing purposes and for computing solutions. Because of the moderate size of the problem and the very efficient BDD-implementation of relations in RELVIEW (see [23]), we even have been able to avoid the use of randomized search heuristics and to compute all existing solutions (even up to isomorphism) or to message that no solution exists. This allowed to detect weak points of the original description. In this situation RELVIEW proved to be an ideal tool for prototyping and validity checks and for the step-wise development of two formal models that finally meet the administration's requirements. The chronologically earlier and also more sophisticated of these models is presented in this paper.

We thank F. Meyer from our university administration for his support and the stimulating discussions and E. Valkema for pointing the administration's timetabling problem to us.

2 Informal Problem Description

The background of the problem is as follows: Presently at our university there exist 34 different fields of study for the undergraduate education of secondary school teachers (and, to be correct, some others professions which corresponds to the former education in these fields of study ending with a Magister degree). According to the examination regulations each student has to select two subjects. Experience with the classical system has shown that all possible combinations can be divided into three categories, viz. the very frequently ones, the less common ones, and those which are hardly ever selected. The goal is to construct a timetable that enables a three years duration of study for combinations of the

first category and a four years duration of study for combinations of the second category. Concretely this means that there are no conflicts between the courses of the two fields of study if they belong to the first category during the entire duration of study and for the second category conflicts at the most appear in one of three years, which enforces a fourth year of study. As a further goal, the number of conflicts should be very small.

To this purpose, the 34 subjects have to be divided into 9 groups, denoted by $A, B, \dots, H, I$, and the groups in turn are divided into three blocks 1, 2 and 3 as shown in the following three tables via the block- and the group-columns:

block	group	year		
		1	2	3
1	A	1	1	1
	B	2	2	2
	C	3	3	3

block	group	year		
		1	2	3
2	D	1	2	3
	E	2	3	1
	F	3	1	2

block	group	year		
		1	2	3
3	G	1	3	2
	H	2	1	3
	I	3	2	1

The meaning of the three year-columns of the tables is as follows. First, each week is divided into three disjoint time slots, denoted by the numbers 1, 2 and 3, and this partitioning remains constant over a long period. For each academic year then each course of the undergraduate-education of secondary school teachers is assigned to a time slot in such a way that all courses of a field of study take place in the same time slot. The table on the left indicates that for the first block this assignment remains constant over three academic years. E.g., every year all courses of a field of study from group A take place in time slot 1. For the other blocks, by contrast, the assignment of courses to time slots cyclically changes, as shown in the remaining two tables. To give also here an example, all courses of a field of study from group D take place in time slot n in year $n, 1 \leq n \leq 3$.

An immediate consequence of the approach is that the duration of study is three years if and only if the two fields of study of the combination belong to different groups of the same block. Four years suffice to take part in the combination's courses if the fields belong to groups of different blocks. Now, from our administration we obtained the classification of the combinations and our task was to compute a function from the fields of study to the groups with the following properties:

- (a) If two fields of study are mapped to the same group, then they form a combination of the third category.
- (b) If two fields of study form a combination of the first category, then their groups belong to the same block.

Both (a) and (b) namely imply that all combinations of the most important first category belong to different groups of the same block. In case that the desired function does not exist, we have been asked to compute at least a partial function for which (a) and (b) hold. Thus, the administration expected to obtain enough information that allows to experiment with the partitioning of the combinations

such that, finally, one is found that allows a solution of the timetabling problem but still is reasonable wrt. the frequency of the combination's choices.

3 Relation-Algebraic Preliminaries

In this section we provide the relation-algebraic material necessary to solve the just informally described problem. For more details concerning relation algebra, see [9] for example.

We denote the set (or type) of all relations with domain X and range Y by $[X \leftrightarrow Y]$ and write $R : X \leftrightarrow Y$ instead of $R \in [X \leftrightarrow Y]$. If the sets X and Y are finite, we may consider R as a Boolean matrix. This interpretation is well suited for many purposes and also one of the possibilities to depict relations in RELVIEW; cf. [11, 3]. Therefore, we use in this paper often matrix notation and terminology. Especially, we speak about rows, columns and entries of relations, and write $R_{x,y}$ instead of $\langle x, y \rangle \in R$ or $x R y$.

We assume the reader to be familiar with the basic operations on relations, viz. $R^\top$ (transposition), $\overline{R}$ (complement), $R \cup S$ (join), $R \cap S$ (meet), RS (composition), $R \subseteq S$ (inclusion), and the special relations $\mathbf{O}$ (empty relation), $\mathbf{L}$ (universal relation) and $\mathbf{I}$ (identity relation). Each type $[X \leftrightarrow Y]$ forms with the operations $\overline{}$, $\cup$, $\cap$, the ordering $\subseteq$ and the constants $\mathbf{O}$ and $\mathbf{L}$ a complete Boolean lattice. Further well-known rules are, e.g., $R^{\top\top} = R$, $\overline{\overline{R}} = R$ and that $R \subseteq S$ implies $R^\top \subseteq S^\top$. The theoretical framework for these rules and many others to hold is that of an (axiomatic, typed) relation algebra. For each type resp. pair / triple of types we have those of the set-theoretic relations as constants and operations of this algebraic structure. The axioms of a relation algebra are the axioms of a complete Boolean lattice for complement, meet, join, ordering, the empty and universal relation, the associativity and neutrality of identity relations for composition, the equivalence of $QR \subseteq S$, $Q^\top \overline{S} \subseteq \overline{R}$, and $\overline{S} R^\top \subseteq \overline{Q}$ (Schröder rule), and that $R \neq \mathbf{O}$ implies $\mathbf{L} R \mathbf{L} = \mathbf{L}$ (Tarski rule). From the latter axiom we obtain that $\mathbf{L} R \mathbf{L} = \mathbf{L}$ or $\mathbf{L} R \mathbf{L} = \mathbf{O}$ and that

$$R \subseteq S \iff \overline{\mathbf{L}(R \cap \overline{S})\mathbf{L}} = \mathbf{L}. \quad (1)$$

Typing the universal relations of the left-hand side of $\overline{\mathbf{L}(R \cap \overline{S})\mathbf{L}} = \mathbf{L}$ in such a way that the universal relation of the equation's right-hand side has a singleton set $\mathbf{1}$ as domain and range and using the only two relations of $[\mathbf{1} \leftrightarrow \mathbf{1}]$ as model for the Booleans, it is possible to translate every Boolean combination φ of relational inclusions into a relation-algebraic expression e such that φ holds if and only if $e = \mathbf{L}$. This follows from the fact that on $[\mathbf{1} \leftrightarrow \mathbf{1}]$ the relational operations $\overline{}$, $\cup$ and $\cap$ directly correspond to the logical connectives $\neg$, $\vee$ and $\wedge$.

There are some relation-algebraic possibilities to model sets. Our first modeling uses (*column*) *vectors*, which are relations v with $v = v\mathbf{L}$. Since for a vector the range is irrelevant, we consider mostly vectors $v : X \leftrightarrow \mathbf{1}$ with the singleton set $\mathbf{1} = \{\perp\}$ as range and omit in such cases the subscript $\perp$, i.e., write v_x instead of $v_{x,\perp}$. Such a vector can be considered as a Boolean matrix with exactly one

column, i.e., as a Boolean column vector, and *represents* the subset $\{x \in X \mid v_x\}$ of X . Sets of vectors are closed under forming complements, joins, meets and left-compositions Rv . As a consequence, for vectors property [\(II\)](#) simplifies to

$$v \subseteq w \iff \overline{\mathsf{L}(v \cap \overline{w})} = \mathsf{L}. \quad (2)$$

With $R^{(y)}$ we denote the y -th column of $R : X \leftrightarrow Y$. I.e., $R^{(y)}$ has type $[X \leftrightarrow \mathbf{1}]$ and for all $x \in X$ are $R_x^{(y)}$ and $R_{x,y}$ equivalent. To compare the columns of two relations R and S with the same domain, we use the right-residual $R \setminus S = \overline{R^\top S}$. Then for all y, y' we have $(R \setminus S)_{y,y'}$ if and only if $R^{(y)} \subseteq S^{(y')}$.

A non-empty vector v is a *point* if $vv^\top \subseteq \mathsf{I}$, i.e., it is *injective*. This means that it represents a singleton subset of its domain or an element from it if we identify a singleton set $\{x\}$ with the element x . In the matrix model, hence, a point $v : X \leftrightarrow \mathbf{1}$ is a Boolean column vector in which exactly one entry is 1.

As a second way we will apply the relation-level equivalents of the set-theoretic symbol $\in$, that is, *membership-relations* $\mathsf{M} : X \leftrightarrow 2^X$. These specific relations are defined by demanding for all elements $x \in X$ and sets $Y \in 2^X$ that $\mathsf{M}_{x,Y}$ iff $x \in Y$. A simple Boolean matrix implementation of membership-relations requires an exponential number of bits. However, in [\[2,3\]](#) an implementation of $\mathsf{M} : X \leftrightarrow 2^X$ using BDDs is presented, where the number of vertices is linear in the size of the base set X . This implementation is part of RELVIEW.

Finally, we will use injective functions for modeling sets. Given an injective function $\iota : Y \rightarrow X$, we may consider Y as a subset of X by identifying it with its image under ι . If Y is actually a subset of X and ι is given as a relation of type $[Y \leftrightarrow X]$ such that $\iota_{y,x}$ iff $y = x$ for all $y \in Y$ and $x \in X$, then the vector $\iota^\top \mathsf{L} : X \leftrightarrow \mathbf{1}$ represents Y as a subset of X in the sense above. Clearly, the transition in the other direction is also possible, i.e., the generation of a relation $\mathit{inj}(v) : Y \leftrightarrow X$ from the vector representation $v : X \leftrightarrow \mathbf{1}$ of the subset Y of X such that for all $y \in Y$ and $x \in X$ we have $\mathit{inj}(v)_{y,x}$ iff $y = x$. A combination of such relations with membership-relations allows a *column-wise representation* of sets of subsets. More specifically, if the vector $v : 2^X \leftrightarrow \mathbf{1}$ represents a subset $\mathcal{S}$ of 2^X in the sense above, then for all $x \in X$ and $Y \in \mathcal{S}$ we get the equivalence of $(\mathsf{M} \mathit{inj}(v)^\top)_{x,Y}$ and $x \in Y$. This means that the elements of $\mathcal{S}$ are represented precisely by the columns of the relation $\mathsf{M} \mathit{inj}(v)^\top : X \leftrightarrow \mathcal{S}$.

Given a product $X \times Y$, there are two projections which decompose a pair $u = \langle u_1, u_2 \rangle$ into its first component u_1 and its second component u_2 . (Throughout this paper pairs u are assumed to be of the form $\langle u_1, u_2 \rangle$.) For a relation-algebraic approach it is very useful to consider instead of these functions the corresponding *projection relations* $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ such that $\pi_{u,x}$ if and only if $u_1 = x$ and $\rho_{u,y}$ if and only if $u_2 = y$. Projection relations algebraically allow to specify the *parallel composition* $R \parallel S : X \times X' \leftrightarrow Y \times Y'$ of relations $R : X \leftrightarrow Y$ and $S : X' \leftrightarrow Y'$ in such a way that $(R \parallel S)_{u,v}$ is equivalent to R_{u_1,v_1} and S_{u_2,v_2} . We get this property if we define

$$R \parallel S = \pi R \sigma^\top \cap \rho S \tau^\top, \quad (3)$$

with $\pi : X \times X' \leftrightarrow X$ and $\rho : X \times X' \leftrightarrow X'$ as projection relations on $X \times X'$ and $\sigma : Y \times Y' \leftrightarrow Y$ and $\tau : Y \times Y' \leftrightarrow Y'$ as projection relations on $Y \times Y'$.

We end this section with two mappings which establish a Boolean lattice isomorphism between the two Boolean lattices $[X \leftrightarrow Y]$ and $[X \times Y \leftrightarrow \mathbf{1}]$. The direction from $[X \leftrightarrow Y]$ to $[X \times Y \leftrightarrow \mathbf{1}]$ is given by the isomorphism vec , where

$$\text{vec}(R) = (\pi R \cap \rho) \mathbf{L}, \quad (4)$$

and that from $[X \times Y \leftrightarrow \mathbf{1}]$ to $[X \leftrightarrow Y]$ by the inverse isomorphism rel , where

$$\text{rel}(v) = \pi^T(\rho \cap v \mathbf{L}^T). \quad (5)$$

In (4) and (5) $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ are projection relations and $\mathbf{L}$ is a universal vector of type $[Y \leftrightarrow \mathbf{1}]$. Using components these definitions say that $R_{x,y}$ if and only if $\text{vec}(R)_{\langle x,y \rangle}$ and that $v_{\langle x,y \rangle}$ if and only if $\text{rel}(v)_{x,y}$. Decisive for our latter applications is the property

$$\text{vec}(QSR) = (Q \parallel R^T) \text{vec}(S). \quad (6)$$

Two immediate consequences of (6) are the special cases $\text{vec}(QS) = (Q \parallel \mathbf{I}) \text{vec}(S)$ and $\text{vec}(SR) = (\mathbf{I} \parallel R^T) \text{vec}(S)$. Property (6) is proved in [5] using (3) and the relation-algebraic axiomatization of the direct product given e.g., in [9].

4 Relation-Algebraic Timetable Construction

To formalize the problem description of Sect. 2, we assume $\mathfrak{S}$ to denote the set of 34 fields of study, $\mathfrak{G}$ to denote the set of 9 groups and $\mathfrak{B}$ to denote the set of 3 blocks. For modeling the partitioning of groups into blocks, we furthermore assume a relation $D : \mathfrak{G} \leftrightarrow \mathfrak{B}$ such that $D_{g,b}$ if and only if group g belongs to block b . Then the reflexive and symmetric relation $B = DD^T : \mathfrak{G} \leftrightarrow \mathfrak{G}$ fulfils

$$B_{g,g'} \iff g \text{ and } g' \text{ belong to the same block.}$$

And, finally, we assume a specification of the partition of the set of all possible combinations of fields of study into the three categories “very frequently”, “less common” and “hardly ever selected” by two relations $J, N : \mathfrak{S} \leftrightarrow \mathfrak{S}$ such that

$$\begin{aligned} J_{s,s'} &\iff s \neq s' \text{ and } (s, s') \text{ is a combination of the first category} \\ N_{s,s'} &\iff s = s' \text{ or } (s, s') \text{ is a combination of the third category.} \end{aligned}$$

Then, $\overline{J \cup N}$ relates two fields of study if and only if they are different and form a combination of the second category. Note that also J and N are symmetric, J is irreflexive, and N is reflexive. The reflexivity of N is motivated by the informal requirement that the duration of study is three years if and only if the two fields of study of the combination belong to different groups of the same block.

Definition 4.1. The relations $B : \mathfrak{G} \leftrightarrow \mathfrak{G}$, $J : \mathfrak{S} \leftrightarrow \mathfrak{S}$ and $N : \mathfrak{S} \leftrightarrow \mathfrak{S}$ constitute the *input of the university timetabling problem*. $\square$

Having fixed the input of our timetabling problem, now we relation-algebraically specify its output.

Definition 4.2. Given the three input relations $B : \mathfrak{G} \leftrightarrow \mathfrak{G}$, $J : \mathfrak{S} \leftrightarrow \mathfrak{S}$ and $N : \mathfrak{S} \leftrightarrow \mathfrak{S}$, a relation $S : \mathfrak{S} \leftrightarrow \mathfrak{G}$ is a *solution of the university timetabling problem*, if the following inclusions hold:

$$\overline{N}S \subseteq \overline{S} \quad JS \subseteq \overline{S\overline{B}} \quad S^\top S \subseteq \mathbf{I} \quad \mathbf{L} \subseteq SL$$

In case that only the first three inclusions hold, S is called a partial solution. $\square$

The four inclusions of Definition 4.2 are a relation-algebraic formalization of the informal requirements of Sect. 2. In the case of $\overline{N}S \subseteq \overline{S}$ this is shown by the following calculation. It starts with the logical formalization of property (a) of Sect. 2 and transforms it step-by-step into the first inclusion of Definition 4.2, thereby replacing logical constructions by their relational counterparts.

$$\begin{aligned} \forall s, s', g : S_{s,g} \wedge S_{s',g} \rightarrow N_{s,s'} &\iff \neg \exists s, s', g : S_{s,g} \wedge S_{s',g} \wedge \overline{N}_{s,s'} \\ &\iff \neg \exists s, g : S_{s,g} \wedge (\overline{N}S)_{s,g} \\ &\iff \forall s, g : (\overline{N}S)_{s,g} \rightarrow \overline{S}_{s,g} \\ &\iff \overline{N}S \subseteq \overline{S} \end{aligned}$$

In the same way the second inclusion $JS \subseteq \overline{S\overline{B}}$ of Definition 4.2 is obtained from the formalization

$$\forall s, s', g, g' : J_{s,s'} \wedge S_{s,g} \wedge S_{s',g'} \rightarrow B_{g,g'}$$

of property (b) of Sect. 2. The remaining two inclusions of Definition 4.2 relation-algebraically specify S to be a *univalent* (third inclusion) and *total* (fourth inclusion) relation, i.e., to be a *function (in the relational sense)*; see [9] for example) from the fields of study to the groups.

Based on an idea presented in [5], the above non-algorithmic relation-algebraic specification of a solution S of our university timetabling problem now will be reformulated in such a way that instead of S its so-called *corresponding vector* $\text{vec}(S)$ is used. This change of representation, finally, will lead to an algorithmic specification. The following theorem is the key of the approach.

Theorem 4.1. Assume B , J and N as in Definition 4.1, a relation $S : \mathfrak{S} \leftrightarrow \mathfrak{G}$ and a vector $v : \mathfrak{S} \times \mathfrak{G} \leftrightarrow \mathbf{1}$ such that $v = \text{vec}(S)$. Then S is a solution of the university timetabling problem if and only if the following inclusions hold:

$$(\overline{N} \parallel \mathbf{I})v \subseteq \overline{v} \quad (J \parallel \mathbf{I})v \subseteq \overline{(\mathbf{I} \parallel \overline{B})v} \quad (\mathbf{I} \parallel \overline{\mathbf{I}})v \subseteq \overline{v} \quad \mathbf{L} \subseteq \pi^\top v$$

In the last inclusion $\pi : \mathfrak{S} \times \mathfrak{G} \leftrightarrow \mathfrak{S}$ is the first projection relation of $\mathfrak{S} \times \mathfrak{G}$.

Proof. We show that for all n , $1 \leq n \leq 4$, the n -th inclusion of Definition 4.2 is equivalent to the n -th inclusion of the theorem. We start with the case $n = 1$:

$$\begin{aligned} \overline{N}S \subseteq \overline{S} &\iff \text{vec}(\overline{N}S) \subseteq \text{vec}(\overline{S}) && \text{vec isomorphism} \\ &\iff (\overline{N} \parallel \mathbf{I})\text{vec}(S) \subseteq \text{vec}(\overline{S}) && \text{due to (6)} \\ &\iff (\overline{N} \parallel \mathbf{I})\text{vec}(S) \subseteq \overline{\text{vec}(S)} && \text{vec isomorphism} \\ &\iff (\overline{N} \parallel \mathbf{I})v \subseteq \overline{v} && v = \text{vec}(S) \end{aligned}$$

The equivalence of the second inclusions is shown as follows:

$$\begin{aligned}
JS \subseteq \overline{S\overline{B}} &\iff \text{vec}(JS) \subseteq \overline{\text{vec}(S\overline{B})} && \text{vec isomorphism} \\
&\iff \text{vec}(JS) \subseteq \overline{\text{vec}(S\overline{B})} && \text{vec isomorphism} \\
&\iff (J \parallel \mathbf{I})\text{vec}(S) \subseteq \overline{(\mathbf{I} \parallel \overline{B}^T)\text{vec}(S)} && \text{due to (6)} \\
&\iff (J \parallel \mathbf{I})\text{vec}(S) \subseteq \overline{(\mathbf{I} \parallel \overline{B})\text{vec}(S)} && B \text{ is symmetric} \\
&\iff (J \parallel \mathbf{I})v \subseteq \overline{(\mathbf{I} \parallel \overline{B})v} && v = \text{vec}(S)
\end{aligned}$$

The following calculation shows the equivalence of the two inclusions concerning univalence of S :

$$\begin{aligned}
S^T S \subseteq \mathbf{I} &\iff S\bar{\mathbf{I}} \subseteq \overline{S} && (4.2.1) \text{ of [9]} \\
&\iff \text{vec}(S\bar{\mathbf{I}}) \subseteq \text{vec}(\overline{S}) && \text{vec isomorphism} \\
&\iff (\mathbf{I} \parallel \bar{\mathbf{I}}^T)\text{vec}(S) \subseteq \text{vec}(\overline{S}) && \text{due to (6)} \\
&\iff (\mathbf{I} \parallel \bar{\mathbf{I}}^T)\text{vec}(S) \subseteq \overline{\text{vec}(S)} && \text{vec isomorphism} \\
&\iff (\mathbf{I} \parallel \bar{\mathbf{I}})v \subseteq \bar{v} && \bar{\mathbf{I}} \text{ is symmetric, } v = \text{vec}(S)
\end{aligned}$$

It remains to verify the last inclusions to be equivalent. Here we have:

$$\begin{aligned}
\mathbf{L} \subseteq S\mathbf{L} &\iff \text{vec}(\mathbf{L}) \subseteq \text{vec}(S\mathbf{L}) && \text{vec isomorphism} \\
&\iff \mathbf{L} \subseteq (\mathbf{I} \parallel \mathbf{L}^T)\text{vec}(S) && \text{vec isomorphism, (6)} \\
&\iff \mathbf{L} \subseteq (\pi\pi^T \cap \rho\mathbf{L}^T\rho^T)\text{vec}(S) && \text{due to (3)} \\
&\iff \mathbf{L} \subseteq (\pi\pi^T \cap \mathbf{L})\text{vec}(S) && \rho \text{ is total} \\
&\iff \mathbf{L} \subseteq \pi\pi^T v && v = \text{vec}(S) \\
&\iff \mathbf{L} \subseteq \pi^T v
\end{aligned}$$

The direction “ $\Rightarrow$ ” of the last step follows from the surjectivity and univalence of π since this implies $\mathbf{L} = \pi^T \mathbf{L} \subseteq \pi^T \pi \pi^T v \subseteq \mathbf{I} \pi^T v = \pi^T v$, and the direction “ $\Leftarrow$ ” is a consequence of the totality of π , since $\mathbf{L} \subseteq \pi \mathbf{L} \subseteq \pi \pi^T v$. $\square$

Now, we are in a position to present a relation-algebraic expression that depends on a vector v and evaluates to the universal relation of $[\mathbf{1} \leftrightarrow \mathbf{1}]$ if and only if v represents a solution of our timetabling problem. In the equation of the following theorem this expression constitutes the left-hand side.

Theorem 4.2. Assume again B, J, N, S, v and π as in Theorem 4.1. Then S is a solution of the university timetabling problem if and only if

$$\overline{\mathbf{L}((\overline{N} \parallel \mathbf{I})v \cap v) \cup ((J \parallel \mathbf{I})v \cap (\mathbf{I} \parallel \overline{B})v) \cup ((\mathbf{I} \parallel \bar{\mathbf{I}})v \cap v) \cup \mathbf{L}\overline{\pi^T v}} = \mathbf{L}.$$

Proof. Property (2) of Sect. 3 implies the following equivalences:

$$\begin{aligned}
(\overline{N} \parallel \mathbf{I})v \subseteq \bar{v} &\iff \overline{\mathbf{L}((\overline{N} \parallel \mathbf{I})v \cap v)} = \mathbf{L} \\
(J \parallel \mathbf{I})v \subseteq \overline{(\mathbf{I} \parallel \overline{B})v} &\iff \overline{\mathbf{L}((J \parallel \mathbf{I})v \cap (\mathbf{I} \parallel \overline{B})v)} = \mathbf{L} \\
(\mathbf{I} \parallel \bar{\mathbf{I}})v \subseteq \bar{v} &\iff \overline{\mathbf{L}((\mathbf{I} \parallel \bar{\mathbf{I}})v \cap v)} = \mathbf{L} \\
\mathbf{L} \subseteq \pi^T v &\iff \overline{\mathbf{L}\overline{\pi^T v}} = \mathbf{L}
\end{aligned}$$

Combining this with Theorem 4.1, we get that S is a solution of our timetabling problem if and only if

$$\overline{\mathbf{L}((\overline{N} \parallel \mathbf{I})v \cap v)} \cap \overline{\mathbf{L}((J \parallel \mathbf{I})v \cap (\mathbf{I} \parallel \overline{B})v)} \cap \overline{\mathbf{L}((\mathbf{I} \parallel \overline{\mathbf{I}})v \cap v)} \cap \overline{\mathbf{L}\pi^{\top}v} = \mathbf{L}$$

Next, we apply a de Morgan law and transform this equation into

$$\overline{\mathbf{L}((\overline{N} \parallel \mathbf{I})v \cap v) \cup \mathbf{L}((J \parallel \mathbf{I})v \cap (\mathbf{I} \parallel \overline{B})v) \cup \mathbf{L}((\mathbf{I} \parallel \overline{\mathbf{I}})v \cap v) \cup \mathbf{L}\pi^{\top}v} = \mathbf{L}.$$

Finally, we replace the universal relation $\mathbf{L} : \mathbf{1} \leftrightarrow \mathfrak{G}$ of $\mathbf{L}\pi^{\top}v$ by a composition $\mathbf{L}\mathbf{L}$, where the first $\mathbf{L}$ has type $[\mathbf{1} \leftrightarrow \mathfrak{G} \times \mathfrak{G}]$ and the second $\mathbf{L}$ has type $[\mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathfrak{G}]$. This adaption of types allows to apply a distributivity law, which yields the desired result. $\square$

Considering v as variable, the left-hand side of the equation of Theorem 4.2 leads to the following mapping Φ on relations, where the first $\mathbf{L}$ has type $[\mathbf{1} \leftrightarrow \mathfrak{G} \times \mathfrak{G}]$, the second $\mathbf{L}$ has type $[\mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathfrak{G}]$ and X is the name of the variable.

$$\Phi(X) = \overline{\mathbf{L}(((\overline{N} \parallel \mathbf{I})X \cap X) \cup ((J \parallel \mathbf{I})X \cap (\mathbf{I} \parallel \overline{B})X) \cup ((\mathbf{I} \parallel \overline{\mathbf{I}})X \cap X) \cup \mathbf{L}\pi^{\top}X)}$$

When applied to a vector $v : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathbf{1}$, this mapping returns $\mathbf{L} : \mathbf{1} \leftrightarrow \mathbf{1}$ if and only if v corresponds to a solution of the university timetabling problem and $\mathbf{0} : \mathbf{1} \leftrightarrow \mathbf{1}$ otherwise. A specific feature of Φ is that it is defined using the variable X , constant relations, complements, joins, meets and left-compositions only. Hence, it is a *vector predicate* in the sense of [5]. With the aid of the membership-relation $\mathbf{M} : \mathfrak{G} \times \mathfrak{G} \leftrightarrow 2^{\mathfrak{G} \times \mathfrak{G}}$ we, therefore, obtain a vector

$$t = \Phi(\mathbf{M})^{\top} : 2^{\mathfrak{G} \times \mathfrak{G}} \leftrightarrow \mathbf{1} \quad (7)$$

such that t_x if and only if the x -column of $\mathbf{M}$ (considered as a vector) corresponds to a solution of our timetabling problem. From (7) a column-wise representation of all vectors which correspond to a solution of our timetabling problem may be obtained using the technique described in Sect. 3. But t also allows to compute a (or even all) single solution(s) in the sense of Definition 4.2. The procedure is rather simple: First, a point $p \subseteq t$ is selected. Because of the above property, the vector $\mathbf{M}p : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathbf{1}$ corresponds to a solution of our timetabling problem. Now, the solution itself is obtained as $\text{rel}(\mathbf{M}p) : \mathfrak{G} \leftrightarrow \mathfrak{G}$.

Each of the relational functions we have presented so far easily can be translated into the programming language of RELVIEW. Using the tool, we have solved the original problem posed to us by the university administration. The input and output relations are too big to be presented here. Therefore, in the following we consider a much smaller example to demonstrate our approach.

Example 4.1. We consider a set $\mathfrak{S}$ of only 10 subjects, namely mathematics (Ma), german (Ge), english (En), history (Hi), physics (Ph), chemistry (Che), biology (Bio), geography (Geo), arts (Ar) and physical education (Pe), which have to be distributed to the six groups A, B, C, D, E and F . The groups are

divided into the blocks 1 and 2 via a relation D and this immediately leads to the relation $B = DD^T : \mathfrak{G} \leftrightarrow \mathfrak{G}$ that specifies whether two groups belong to the same block. As RELVIEW-matrices D and B look as follows:

$$D = \begin{array}{c} \begin{array}{cc} & 1 \quad 2 \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{array}{cc} \blacksquare & \\ \blacksquare & \\ \blacksquare & \\ \blacksquare & \\ \blacksquare & \\ \blacksquare & \end{array} \end{array} & B = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{array}{cccccc} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{array} \end{array} \end{array}$$

We further consider the first two tables of Sect. 2, that assign one time slot to every group $A, B, \dots, F$ for each of the three years. The three relations J, N and B , where J and N are shown in the following pictures as RELVIEW-matrices, constitute the input of our exemplary timetabling problem. From the pictures we see e.g., that mathematics and physics constitute an often selected combination and history and chemistry are hardly ever combined.

$$J = \begin{array}{c} \begin{array}{cccccccc} & Ma & Ge & En & Hi & Ph & Che & Bio & Geo & Ar & Pe \\ \begin{array}{c} Ma \\ Ge \\ En \\ Hi \\ Ph \\ Che \\ Bio \\ Geo \\ Ar \\ Pe \end{array} & \begin{array}{cccccccc} \blacksquare & & & & & & & & & & \\ & \blacksquare & & & & & & & & & \\ & & \blacksquare & & & & & & & & \\ & & & \blacksquare & & & & & & & \\ & & & & \blacksquare & & & & & & \\ & & & & & \blacksquare & & & & & \\ & & & & & & \blacksquare & & & & \\ & & & & & & & \blacksquare & & & \\ & & & & & & & & \blacksquare & & \\ & & & & & & & & & \blacksquare & \\ & & & & & & & & & & \blacksquare \end{array} \end{array} & N = \begin{array}{c} \begin{array}{cccccccc} & Ma & Ge & En & Hi & Ph & Che & Bio & Geo & Ar & Pe \\ \begin{array}{c} Ma \\ Ge \\ En \\ Hi \\ Ph \\ Che \\ Bio \\ Geo \\ Ar \\ Pe \end{array} & \begin{array}{cccccccc} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{array} \end{array} \end{array}$$

We have used RELVIEW to generate the membership-relation $M : \mathfrak{S} \times \mathfrak{G} \leftrightarrow 2^{\mathfrak{S} \times \mathfrak{G}}$ of size 60×2^{60} for this example and to determine then the vector $t = \Phi(M)^T$ of length 2^{60} by translating the definition of Φ into its programming language. The tool showed that t has 144 1-entries, which means that there are exactly 144 solutions for the given problem, represented by 144 columns of M . Selecting a point $p \subseteq t$ and defining v as composition Mp , a vector of type $[\mathfrak{S} \times \mathfrak{G} \leftrightarrow \mathbf{1}]$ and its corresponding relation $S = rel(v) : \mathfrak{S} \leftrightarrow \mathfrak{S}$ have been computed such that the latter is a solution of our timetabling problem. Here is its RELVIEW-picture:

$$S = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \begin{array}{c} Ma \\ Ge \\ En \\ Hi \\ Ph \\ Che \\ Bio \\ Geo \\ Ar \\ Pe \end{array} & \begin{array}{cccccc} \blacksquare & & & & & \\ & \blacksquare & & & & \\ & & \blacksquare & & & \\ & & & \blacksquare & & \\ & & & & \blacksquare & \\ & & & & & \blacksquare \\ & & & & & & \blacksquare \\ & & & & & & & \blacksquare \\ & & & & & & & & \blacksquare \\ & & & & & & & & & \blacksquare \end{array} \end{array} \end{array}$$

Using the composition $M inj(t)^T$ we even have been able to compute the list of all solutions, represented as a relation with 60 rows and 144 columns. This relation is too large to be depicted here. $\square$

5 Computing Solutions Up to Isomorphism

If our timetabling problem is solvable, there often exist a large number of solutions. To be able to evaluate and compare the solutions, it is useful to examine

them for isomorphism and consider only one solution of a large set of very similar ones. In this section we will show how this can be achieved. First we will present a reasonable definition of isomorphism between solutions, based on the sets of combinable and restricted combinable pairs of subjects. For a given solution S , we call two subjects *combinable*, if they can be studied without overlappings, which means that S assigns the subjects to different groups of the same block. Two subjects that are assigned to groups of different blocks are called *restricted combinable*. The following lemma gives relation-algebraic expressions that specify the combinable and restricted combinable pairs of subjects, respectively.

Lemma 5.1. Assume the input relation $B : \mathfrak{G} \leftrightarrow \mathfrak{G}$ and the solution $S : \mathfrak{S} \leftrightarrow \mathfrak{G}$ of our timetabling problem and define the relations $co(S)$ and $reco(S)$ of type $[\mathfrak{S} \leftrightarrow \mathfrak{S}]$ as follows:

$$co(S) = S(B \cap \bar{I})S^T \quad reco(S) = S\bar{B}S^T$$

Then it holds for all $s, s' \in \mathfrak{S}$ that $co(S)_{s,s'}$ if and only if s and s' are combinable and $reco(S)_{s,s'}$ if and only if s and s' are restricted combinable.

Proof. Given arbitrary elements $s, s' \in \mathfrak{S}$, it holds that

$$\begin{aligned} s \text{ and } s' \text{ are combinable} &\iff \exists g, g' : S_{s,g} \wedge S_{s',g'} \wedge g \neq g' \wedge B_{g,g'} \\ &\iff \exists g, g' : S_{s,g} \wedge S_{s',g'} \wedge (\bar{I} \cap B)_{g,g'} \\ &\iff (S(B \cap \bar{I})S^T)_{s,s'} \end{aligned}$$

and in a similar way the second claim is verified. $\square$

Based on the above relational mappings co and $reco$, we are now in the position formally to define our notion of isomorphism.

Definition 5.1. Two solutions S and S' of the university timetabling problem are called *isomorphic* if $co(S) = co(S')$ and $reco(S) = reco(S')$. In this case we write $S \cong S'$. $\square$

Recall that a relation P for which domain and range coincide is called a permutation if and only if P as well as its transpose P^T are functions in the relational sense. As we will see later, we can use block-preserving permutation relations to create isomorphic solutions from a given solution of our timetabling problem. This specific kind of permutation relations is introduced as follows.

Definition 5.2. Given B as in Lemma 5.1, we call a permutation relation $P : \mathfrak{S} \leftrightarrow \mathfrak{S}$ *block-preserving* if $B \subseteq PBP^T$. $\square$

In words the inclusion $B \subseteq PBP^T$ means that if two groups belong to the same block, then this holds for their images under the permutation relation, too. The following theorem clarifies the relationship between isomorphism of solutions and block-preserving permutation relations. Its proof is omitted due to space restrictions. The first part is an immediate consequence of the definitions, the more complicated proof of the second part will be published in the forthcoming Ph.D. thesis [6].

Theorem 5.1. a) If the relation S is a solution of the university timetabling problem and P a block-preserving permutation relation, then SP is also a solution and $S \cong SP$.

b) For two solutions S and S' we have $S \cong S'$ if and only if there exists a block-preserving permutation relation P such that $S' = SP$. $\square$

To determine the set of all solutions that are isomorphic to a given solution S , we start with the following theorem. It states a relation-algebraic expression that depends on a vector v and evaluates to the $\mathbf{L}$ of type $[1 \leftrightarrow 1]$ if and only if v is the corresponding vector of a block-preserving permutation relation.

Theorem 5.2. Let B be as in Lemma 5.1. Furthermore, assume $P : \mathfrak{G} \leftrightarrow \mathfrak{G}$ and and a vector $v : \mathfrak{G} \times \mathfrak{G} \leftrightarrow 1$ such that $v = \text{vec}(P)$. Then P is a block-preserving permutation relation if and only if

$$\overline{\mathbf{L}(\overline{\pi^\top v} \cup \overline{\rho^\top v} \cup (v \cap ((\mathbf{I} \parallel \bar{\mathbf{I}}) \cup (\bar{\mathbf{I}} \parallel \mathbf{I}) \cup (B \parallel \bar{B}))v))} = \mathbf{L},$$

where $\pi : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathfrak{G}$ and $\rho : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathfrak{G}$ are the projection relations of $\mathfrak{G} \times \mathfrak{G}$.

Proof. Like in Theorem 4.1 we can show the following two equivalences by combining the assumption $v = \text{vec}(P)$ with the properties (2) and (6):

$$\begin{aligned} P \text{ injective} &\iff \overline{\mathbf{L}((\bar{\mathbf{I}} \parallel \mathbf{I})v \cap v)} = \mathbf{L} \\ P \text{ surjective} &\iff \overline{\mathbf{L}\rho^\top v} = \mathbf{L} \end{aligned}$$

Using additionally the relation-algebraic equations for specifying univalence and totality of relations given in the proof of Theorem 4.2 for P and its corresponding vector v , we obtain that P is a permutation relation if and only if

$$\overline{\mathbf{L}((\mathbf{I} \parallel \bar{\mathbf{I}})v \cap v)} \cap \overline{\mathbf{L}\pi^\top v} \cap \overline{\mathbf{L}((\bar{\mathbf{I}} \parallel \mathbf{I})v \cap v)} \cap \overline{\mathbf{L}\rho^\top v} = \mathbf{L}.$$

Supposing this equation to hold, now we are able to calculate as follows:

$$\begin{array}{ll} B \subseteq PBP^\top &\iff BP \subseteq PB && P \text{ function} \\ &\iff B^\top P \subseteq PB && B \text{ symmetric} \\ &\iff B\bar{P}\bar{B} \subseteq \bar{P} && \text{Schröder rule} \\ &\iff BP\bar{B} \subseteq \bar{P} && P \text{ function, (4.2.4) of [9]} \\ &\iff \text{vec}(BP\bar{B}) \subseteq \text{vec}(\bar{P}) && \text{vec isomorphism} \\ &\iff (B \parallel \bar{B}^\top)\text{vec}(P) \subseteq \overline{\text{vec}(P)} && \text{vec isomorphism, (6)} \\ &\iff (B \parallel \bar{B}^\top)v \subseteq \bar{v} && v = \text{vec}(P) \\ &\iff \overline{\mathbf{L}(v \cap (B \parallel \bar{B})v)} = \mathbf{L} && \text{due to (2)} \end{array}$$

If we intersect the left-hand side of the last equation of this derivation with the left-hand side of the above equation, we get that P is a block-preserving permutation relation if and only if

$$\overline{\mathbf{L}((\mathbf{I} \parallel \bar{\mathbf{I}})v \cap v)} \cap \overline{\mathbf{L}\pi^\top v} \cap \overline{\mathbf{L}((\bar{\mathbf{I}} \parallel \mathbf{I})v \cap v)} \cap \overline{\mathbf{L}\rho^\top v} \cap \overline{\mathbf{L}(v \cap (B \parallel \bar{B})v)} = \mathbf{L}.$$

The last steps of the proof are rather the same as in the case of Theorem 4.2. We use a de Morgan law, introduce two universal relations for type adaption and apply commutativity of join and a distributivity law. $\square$

Like in Sect. 4, from Theorem 5.2 we immediately obtain the following mapping Ψ on relations that is defined using the variable X , constant relations, complements, joins, meets and left-compositions only:

$$\Psi(X) = \overline{\text{L}(\text{L} \pi^T \overline{X} \cup \text{L} \rho^T \overline{X} \cup (X \cap ((\text{I} \parallel \bar{\text{I}}) \cup (\bar{\text{I}} \parallel \text{I}) \cup (B \parallel \overline{B})))X)}$$

As a consequence, the application of the vector predicate Ψ to the membership-relation $M : \mathfrak{G} \times \mathfrak{G} \leftrightarrow 2^{\mathfrak{G} \times \mathfrak{G}}$ and a transposition of the result yield a vector

$$b = \Psi(M)^T : 2^{\mathfrak{G} \times \mathfrak{G}} \leftrightarrow 1 \quad (8)$$

that specifies exactly those columns of M which are corresponding vectors of block-preserving permutation relations. According to the technique of Sect. 3, hence, a column-wise representation of the set $\mathcal{P}$ of all block-preserving permutation relations (as a subset of all relations on $\mathfrak{G}$) is given by the relation

$$E = M \text{ inj}(b)^T : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathcal{P}. \quad (9)$$

To be more precise, the mapping $P \mapsto \text{vec}(P)$ constitutes a one-to-one correspondence between $\mathcal{P}$ and the set of all columns of E (where each column is considered as a vector of type $[\mathfrak{G} \times \mathfrak{G} \leftrightarrow 1]$). In the remainder of the section we show how the relation of (9) can be used to compute the set of all solutions isomorphic to a given solution S . The decisive property is presented in the next theorem. It states a relation-algebraic expression for the column-wise representation of all solutions isomorphic to S , where however, in contrast to the notion introduced in Sect. 3, multiple occurrences of columns are allowed. In the proof we use the notation $R^{(x)}$ for the x -th column of R as introduced in Sect. 3.

Theorem 5.3. Assume $S : \mathfrak{G} \leftrightarrow \mathfrak{G}$ to be a solution of the university timetabling problem and the relation I_S to be defined as

$$I_S = (S \parallel \text{I})E : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathcal{P}.$$

Then every $x \in \mathcal{P}$ leads to a solution $\text{rel}(I_S^{(x)})$ such that $\text{rel}(I_S^{(x)}) \cong S$ and for every solution S' with $S' \cong S$ there exists $x \in \mathcal{P}$ such that $\text{vec}(S') = I_S^{(x)}$.

Proof. To prove the first statement, we assume $x \in \mathcal{P}$. Since $I_S = (S \parallel \text{I})E$, we have $I_S^{(x)} = (S \parallel \text{I})E^{(x)}$. Now, the above mentioned one-to-one correspondence between the set $\mathcal{P}$ and the set of all columns of E shows the existence of a block-preserving permutation relation $P : \mathfrak{G} \leftrightarrow \mathfrak{G}$ fulfilling $E^{(x)} = \text{vec}(P)$, i.e.,

$$I_S^{(x)} = (S \parallel \text{I})E^{(x)} = (S \parallel \text{I})\text{vec}(P) = \text{vec}(SP)$$

because of property (6). This equation in turn leads to

$$\text{rel}(I_S^{(x)}) = \text{rel}(\text{vec}(SP)) = SP$$

and, finally, Theorem 5.1 a) shows the desired result.

For a proof of the second claim, we start with a solution S' such that $S' \cong S$. Then Theorem 5.1 b) yields a block-preserving permutation relation $P : \mathfrak{G} \leftrightarrow \mathfrak{G}$ with $S' = SP$. Next, we apply property (6) and get

$$\text{vec}(S') = \text{vec}(SP) = (S \parallel \mathbf{1}) \text{vec}(P).$$

Since E column-wisely represents the block-preserving permutation relations, there exists a column $E^{(x)}$ such that $\text{vec}(P) = E^{(x)}$. Combining this with the above result and the definition of I_S yields $\text{vec}(S') = (S \parallel \mathbf{1})E^{(x)} = I_S^{(x)}$. $\square$

Now, we use Theorem 5.3 and describe a procedure for the computation of the set of all solutions of our timetabling problem up to isomorphism. It easily can be implemented in RELVIEW. In a first step, we determine the vector $t : 2^{\mathfrak{G} \times \mathfrak{G}} \leftrightarrow \mathbf{1}$ of (7) that specifies those columns $M : \mathfrak{G} \times \mathfrak{G} \leftrightarrow 2^{\mathfrak{G} \times \mathfrak{G}}$ which correspond to solutions of the timetabling problem, and the relation $E : \mathfrak{G} \times \mathfrak{G} \leftrightarrow \mathcal{P}$ of (9) that does the same for the block-preserving permutation relations. Selecting a point p from t , we then compute a single solution S as described in Sect. 4 and the column-wise representation I_S of all solutions isomorphic to S . With $t' = t \cap (M \setminus I_S)L : 2^{\mathfrak{G} \times \mathfrak{G}} \leftrightarrow \mathbf{1}$ we obtain a vector that specifies all columns of M that correspond to solutions isomorphic to S . This follows from

$$\begin{aligned} (t \cap (M \setminus I_S)L)_x &\iff t_x \wedge \exists y : (M \setminus I_S)_{x,y} && \text{see Sect. 3} \\ &\iff t_x \wedge \exists y : M^{(x)} \subseteq I_S^{(y)} && \text{solutions have same size} \\ &\iff \exists y : M^{(x)} = I_S^{(y)} && \text{Theorem 5.3} \\ &\iff \text{rel}(M^{(x)}) \cong S \end{aligned}$$

for all $x \in 2^{\mathfrak{G} \times \mathfrak{G}}$. By modifying t to $t \cap \bar{t}'$ we can remove all solutions isomorphic to S from t . Successive application of this approach leads to a vector that, finally, represents one element of each set of isomorphic solutions.

Experience has shown that in most cases, the number of solutions can be reduced considerable if we restrict us to non-isomorphic ones. So there exist 1296 block-preserving permutations for the original problem of Sect. 2 with 9 groups and 3 blocks, so that for each solution there are up to 1296 isomorphic solutions. Regarding Example 4.1, where we deal with 2 blocks and 6 groups only, there are 72 block-preserving permutations, and the 144 solutions of the timetabling problem can be reduced to only two solutions which are not isomorphic.

6 Concluding Remarks

Having formalized the timetabling problem posed to us by the administration of our university and having developed a relational algorithm for its solution, we implemented the algorithm in RELVIEW and applied it to the input data. The administration delivered the latter electronically in tabular form and we used a small Java-program to convert these files into RELVIEW's so-called ASCII file-format. Loading the RELVIEW-files into the tool and performing the algorithm.

we obtained the vector t of [\[7\]](#) to be empty. Since this meant that there exists no solution, in accordance with the university administration we changed the three categories of possible combinations slightly and applied the RELVIEW-program to the new relations J and N . Again we got $t = \mathbf{0}$. Repeating this process several times, we finally found a non-empty t . But thus we had changed the categories in such a way that a further perpetuation of the trisection of the combinations seemed inappropriate. So, we decided to drop the category “less common” and to work with the remaining two categories only. This modified approach, finally, led to 32 solutions with only 17% of the combinations in the category “hardly ever selected”. One of the 32 solutions has been chosen by our administration. At present it is discussed in commissions of single departments, the faculties, and the entire university. The ultimate decision about introduction and final form of the timetable depends on the results of these discussions.

During the entire project RELVIEW proved to be an ideal tool for the tasks to be solved. Systematic experiments helped us to get insight into the specific character of the problem and to develop the relation-algebraic formalizations. Because of their concise form it was very easy to adapt the programs of the original model to the new one and to write auxiliary programs for testing and visualization purposes. Particularly with regard to the above mentioned step-wise change of the categories we have used a small RELVIEW-program that enumerates all maximum cliques of an undirected graph since the existence of large cliques typically prevented a solution of our timetabling problem.

References

1. Behnke, R., et al.: RELVIEW – A system for calculation with relations and relational programming. In: Astesiano, E. (ed.) ETAPS 1998 and FASE 1998. LNCS, vol. 1382, pp. 318–321. Springer, Heidelberg (1998)
2. Berghammer, R., Leonniuk, B., Milanese, U.: Implementation of relation algebra using binary decision diagrams. In: de Swart, H. (ed.) RelMiCS 2001. LNCS, vol. 2561, pp. 241–257. Springer, Heidelberg (2002)
3. Berghammer, R., Neumann, F.: RELVIEW – An OBDD-based Computer Algebra system for relations. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2005. LNCS, vol. 3718, pp. 40–51. Springer, Heidelberg (2005)
4. Gross, J.L., Yellen, J. (eds.): Handbook of graph theory. CRC Press, Boca Raton (2003)
5. Kehden, B.: Evaluating sets of search points using relational algebra. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 266–280. Springer, Heidelberg (2006)
6. Kehden, B.: Vectors and vector predicates and their use in the development of relational algorithms (in German). Ph.D. thesis, Univ. of Kiel (to appear, 2008)
7. Schmidt, G., Ströhlein, T.: Some aspects in the construction of timetables. In: Rosenfeld, J.L. (ed.) Proc. IFIP Congress 1974, pp. 516–520. North Holland, Amsterdam (1974)
8. Schmidt, G., Ströhlein, T.: A Boolean matrix iteration in timetable construction. *Lin. Algebra and Applications* 15, 27–51 (1976)
9. Schmidt, G., Ströhlein, T.: Relations and graphs. Springer, Heidelberg (1993)
10. Tarski, A.: On the calculus of relations. *J. Symbolic Logic* 6, 73–89 (1941)

A Relation Algebraic Semantics for a Lazy Functional Logic Language^{*}

Bernd Braßel and Jan Christiansen

Department of Computer Science
University of Kiel, 24098 Kiel, Germany
{bbr,jac}@informatik.uni-kiel.de

Abstract. We propose a relation algebraic semantics along with a concrete model for lazy functional logic languages. The resulting semantics provides several interesting advantages over former approaches for this class of languages. On the one hand, the high abstraction level of relation algebra allows equational reasoning leading to concise proofs about functional logic programs. On the other hand the proposed approach features, in contrast to former approaches with a comparable level of abstraction, an explicit modeling of sharing. The latter property gives rise to the expectation that the presented framework can be used to clarify notions currently discussed in the field of functional logic languages, like constructive negation, function inversion and encapsulated search. All of these topics have proved to involve subtle problems in the context of sharing and laziness in the past.

1 Introduction and Motivation

In contrast to traditional imperative programming languages, declarative languages provide a higher and more abstract level of programming, see [10] for a recent survey. There are two main streams of research concerning declarative languages: logic and functional programming. Since the early nineties a third stream of research aims to combine the advantages of both paradigms and create functional logic programming languages. One of the resulting languages is called Curry [10] which is used in the examples of this work. By now the research field of functional logic programming languages is well developed, including several approaches to provide denotational semantics for functional logic languages [11, 9, 12] to enable mathematical reasoning about programs. However, recent works document that there are still basic questions which have not been answered satisfactorily yet. These questions concern for instance the integration of logic search such that results from different branches of a search space can be collected or compared. Such a comparison is essential, e.g., to implement optimization problems employing the built-in search of functional logic languages. As discussed in [6] approaches to integrate logic search in this way are either not

^{*} This work has been partially supported by the German Research Council (DFG) under grant Ha 2457/5-2.

expressive enough [13] or compromise important properties [3]. Another question concerns the notion of inversion. Especially in the context of lazy evaluation it is up to now not at all clear what the inversion of a functional logic operation should be. The programming language Curry provides a feature called function patterns that implements a kind of inversion [2]. Right now there is only an operational semantics describing this feature but no denotational one. Furthermore, the semantic approaches employed in the area do often lead to lengthy and very technical proofs which often do not convey the central proof idea well, see for instance the proofs in [6]. This is mostly due to the fact that a special aspect called *sharing* has been abstracted from in high level approaches to program semantics [9]. Adding this aspect in the way proposed in [1] or [8] increases the level of technical detail considerably. The need for improvement in this regard is documented by a recent approach to add sharing in a less technical way [12].

In this paper we present a new approach to a denotational semantics for functional logic languages employing relation algebra. Unfortunately it is beyond the scope of this paper to demonstrate that the problems stated above can indeed be solved by the presented algebraic methods. However, we are optimistic that notions like inversion and the integration of logic search can be given a clear and precise meaning in a relation algebraic framework. Moreover, we think that the relation algebraic representation of sharing is both explicit enough to be fruitfully employed but avoids more technicality than former approaches including [12].

In the remaining paper we give an introduction to functional logic languages (Section 1.1) and relation algebra (Section 1.2). The main Section 2 contains the development of the relation algebraic semantics for functional logic languages, followed by concluding remarks (Section 3).

1.1 Functional Logic Programming Languages

A functional logic program is a *constructor-based term rewriting system*. Terms are inductively built from a *signature* Σ , i.e., a set of symbols with corresponding arity and a set of variables $\mathcal{X}$. In a *constructor-based term rewriting system*, the signature is partitioned into two sets, the *operator* and *constructor* symbols.

Definition 1 ((Constructor-Based) Signature, Term, Substitution). *A signature Σ is a set of symbols with associated arity. A constructor-based signature Σ additionally features a disjoint partition $\Sigma = op(\Sigma) \cup cons(\Sigma)$ and we call $op(\Sigma)$ the operator and $cons(\Sigma)$ the constructor symbols of Σ . By convention, we write s^n to denote that the symbol s has the associated arity n but may omit the arity of a symbol when convenient. Generally, we use f, g, h for operator symbols, c, d for constructor symbols and s for an arbitrary symbol.*

*Let $\mathcal{X}$ be a set of variables. Then the set of terms over Σ and $\mathcal{X}$ is denoted by $T_\Sigma(\mathcal{X})$ and we refer to the set of variables contained in a term t as $var(t)$. Furthermore, a term t is *linear* if every variable in $var(t)$ appears only once in t . Let σ be a mapping from variables to terms. Then the homomorphic extension of σ with respect to term structure is called a *substitution* and we identify the*

substitution with that mapping. Substitutions are denoted by σ and we call σ a constructor-substitution if it maps variables to a subset of $T_{cons(\Sigma)}(\mathcal{X})$ only.

In the following we assume without loss of generality Σ to be fixed and that there is no symbol with name $\mathbf{U}$ in Σ .

Example 1 (Data Declarations). Curry is a statically typed language and constructors are always introduced with a corresponding type. A new type along with its constructors is introduced by a **data** declaration. The following two declarations define a Boolean type and a data type of polymorphic lists. A Boolean type has two nullary constructors **True** and **False**. The list type has a nullary constructor **Nil** representing the empty list and a binary constructor **Cons**. The “**a**” in the second declaration denotes that **List a** is a polymorphic type. That is, it represents lists that contain elements of arbitrary but equal types.

```
data Bool = True | False      data List a = Nil | Cons a (List a)
```

In the semantics we abstract from the different types of a Curry program and associate each symbol with its arity only, cf. Definition [1](#).

A functional logic program is a *term rewriting system*, i.e., a set of equations which are used from left to right to *evaluate expressions*. In the constructor-based setting, the left hand sides of the equations have a special form. They are all rooted by operator symbols, whereas the inner terms, called *patterns*, are linear terms built from constructors and variables only.

Definition 2 (Constructor-Based Term Rewriting System). *Let Σ be a constructor-based signature. A constructor-based Σ -term rewriting system is a set of rules of the form $f\ t_1 \dots t_n = r$ where $f^n \in \Sigma$, $i \in \{1 \dots n\}$, $t_i \in T_{cons(\Sigma)}(\mathcal{X})$, t_i linear and $r \in T_{\Sigma}(\mathcal{X})$.*

Example 2 (Declaring Operations). In Curry the Boolean negation **not** and the partial function **head**, retrieving the first element of a list, are defined by:

```
not True  = False      head (Cons x xs) = x
not False = True
```

By convention, operator symbols are written lower case while constructors start with a capital letter. Curry is a statically typed language, but there is type inference. Thus, the type signatures **not** :: **Bool** -> **Bool** and **head** :: **List a** -> **a** can be added by the user, optionally. Operations with more than one argument like the Boolean if and only if **iff** :: **Bool** -> **Bool** -> **Bool** can be written as:

```
iff True  x = x
iff False x = not x
```

In Curry overlapping left hand sides lead to non-determinism. For example the operation **coin** :: **Bool** non-deterministically evaluates to **True** or **False**.

```
coin = True
coin = False
```


To define what evaluating an expression means we first define the notion of a context with a hole. This allows a concise notation of replacing a sub-term in a given term.

Definition 3 (Context). *Let Σ be a signature. Contexts (with one hole) are defined to be either a hole $\square$ or to be of the form $(s\ t_1\ \dots\ \mathcal{C}\ \dots\ t_n)$ where $\mathcal{C}$ is a context, $s^{n+1} \in \Sigma$ and for each $i \in \{1 \dots n\}$ t_i is in $T_\Sigma(\mathcal{X})$. The application of a context $\mathcal{C}$ to a term t , written as $\mathcal{C}[t]$ is defined inductively by $\square[t] = t$ and $(s\ t_1\ \dots\ \mathcal{C}\ \dots\ t_n)[t] = (s\ t_1\ \dots\ \mathcal{C}[t]\ \dots\ t_n)$.*

Example 3. Two examples of context applications:

$$(\text{iff } \square\ \text{False})[\text{True}] = \text{iff True False} \qquad \square[\text{False}] = \text{False}$$

Next we will see how these two applications are put together to form the evaluation of the expression `(iff (not False) False)` in the context of Example 2.

If we wanted to define a *strict* functional logic language, we would be done by simply stating the following rule.

Definition 4 (Operation Unfolding). *Let P be a program containing the rule $f\ t_1 \dots t_n = e$, σ a constructor-substitution and $\mathcal{C}$ a context. Then an unfolding step of f is of the form $\mathcal{C}[f\ \sigma(t_1) \dots \sigma(t_n)] \rightsquigarrow \mathcal{C}[\sigma(e)]$.*

Example 4. A sequence of unfolding steps using the declarations of Example 2:

$$\text{iff (not False) False} \rightsquigarrow \text{iff True False} \rightsquigarrow \text{False}$$

In a *strict* functional (logic) language, the arguments of a function call are evaluated before the function is applied. That is why strictness is also referred to as *call-by-value*. The dual conception *call-by-name* allows to unfold a function call before the arguments are fully evaluated. Call-by-name allows a more expressive style of programming [11] and every general purpose language has at least one construct which is partially applied by name and not by value, e.g., if-then-else.

Example 5 (Potentially Infinite Data Structure). One of the advantages of call-by-name is the possibility to compute with (potentially) infinite objects. For example, the operation `true`s declared as follows yields a list of arbitrary length.

```
true
```

In a call-by-name language, the expression `(head true`s) evaluates to `True` while in a call-by-value language, evaluating that expression would not terminate.

A pure call-by-name semantics has a severe disadvantage which directly leads to the concept of *laziness* (call-by-need). This disadvantage becomes apparent whenever a function copies one of its argument variables. As the arguments are not fully evaluated before application, copying an argument means doubling the work to evaluate the arguments whenever the value of both copies is needed.

Example 6 (Pure Call-By-Name). Consider the following operation:

```
copy :: Bool -> Bool
copy x = iff x x
```

In a pure call-by-name approach the evaluation of `(copy (head trues))` would induce the following evaluation sequence:

```
copy (head trues) ~> iff (head trues) (head trues)
~> iff (head (True:trues)) (head trues) ~> iff True (head trues)
~> head trues ~> head (True:trues) ~> True
```

Because of being copied the sub-expression `(head trues)` is evaluated twice.

The straight forward solution to omit copying expressions is to copy *references* to expressions only. The resulting approach is called laziness or call-by-need. In most models of such an approach, terms are replaced by directed acyclic graphs. Sub-expressions which are referenced more than once, i.e., nodes with an in-degree ≥ 2 , are called *shared*.

Example 7 (Evaluation with Sharing). With sharing expression `(head trues)` is only evaluated once:

```
copy (head trues) ~> iff          ~> iff          ~> iff          ~> True
                        ↘ ↙          ↘ ↙          ↘ ↙
                (head trues)  (head (True:trues))  True
```

Many approaches model sharing by explicitly adding graph terms or a similar means to express references to expressions [13,12]. We, however, follow [9] and make use of the fact that non-determinism is a more general concept than laziness. The main feature of laziness is that it allows to *not* evaluate certain sub-expressions. By making the choice whether or not to evaluate any expression non-deterministically, the same effect can be achieved. Therefore, laziness can be introduced by adding a (polymorphic) constructor symbol U (for unevaluated) and allow the arbitrary replacement of expressions by U [1].

Definition 5 (Discarding Expressions). *Let $\mathcal{C}$ be a context and t a term. Then a discarding step is of the form $\mathcal{C}[t] \rightarrow \mathcal{C}[\mathsf{U}]$.*

Example 8 (Laziness). Together, unfolding and discarding steps allow the definition and evaluation of potentially infinite data structures:

```
head trues → head (Cons True trues) → head (Cons True U) → True
```

The addition of shared expressions implies an additional design decision for the extension to functional *logic* languages. In a functional logic language a shared expression can non-deterministically evaluate to different values, e.g., in the evaluation of `(copy coin)`. Should there be only one choice for *all* references

¹ Extending a strict language with laziness employing non-determinism is not an option in practice. The traditional techniques employ so called promises or futures along with operations `force` and `delay`. This approach is also followed in [16].

to the expression in this situation or should there be independent choices for *each* such reference? The decision that there is only one choice for all references corresponds to what is known as *call-time choice*; the dual conception is called *run-time choice*. Curry features call-time choice which is reflected by the constraint in Definition 4 that σ has to be a *constructor*-substitution rather than a general substitution.

Example 9 (Call-Time Choice). For the following program along with the declarations of `coin` and `copy` from above the following sequence is valid:

`copy coin` $\rightarrow$ `copy True` $\rightarrow$ `iff True True` $\rightarrow$ `True`

Since the first step requires the variable of the rule of `copy` to be substituted with `coin`, which is not a constructor term, the following sequence is not valid:

`copy coin` $\not\rightarrow$ `iff coin coin` $\rightarrow$ `iff True coin` $\rightarrow$ `iff True False` $\rightarrow$ `False`

Definition 2 does not require variables of the right-hand side of an equation to appear in the left-hand side. Variables appearing in the right-hand side only, called *free variables*, can only be substituted with constructor terms by Definition 4.

Example 10 (Free Variable). In Curry a free variable x appearing in an expression e is introduced by the declaration `let x free in e`, for example:

```
expr :: Bool
expr = let x free in iff x x
```

The possible evaluation sequences for `expr` are:

`expr` $\rightarrow$ `iff x x` $\rightarrow$ `True` `expr` $\rightarrow$ `iff x x` $\rightarrow$ `not False` $\rightarrow$ `True`

1.2 Relation Algebra

We assume the reader to be familiar with the basic concepts of relation algebra and with the basic operations on relations, viz. $R \cup S$ (*union*), $R \cap S$ (*intersection*), and $R \circ S$ (*multiplication*), $R^\top$ (*transposition*), $R \subseteq S$ (*inclusion*), and the special relations $\mathbf{O}$ (*empty relation*), $\mathbf{L}$ (*universal relation*), and $\mathbf{I}$ (*identity relation*). For a detailed introduction to relation algebra see for example [15]. We also give concrete models for some of the relations to provide a better intuition. We write $R : X \leftrightarrow Y$ if R is a concrete relation with domain X and range Y , i.e., a subset of $X \times Y$. We denote an element of $X \times Y$ by $\langle x, y \rangle$.

Furthermore we make use of the projections of a direct product π and ρ and the injections of a direct sum ι_1 and ι_2 . For relations R and S we define their *tupling* $[R, S] := R \circ \pi^\top \cap S \circ \rho^\top$ and their parallel composition $R \parallel S := \pi \circ R \circ \pi^\top \cap \rho \circ S \circ \rho^\top$. In the concrete model π and ρ are the projections of the Cartesian product $X \times Y$ into X and Y respectively. We assume the operator $\times$ to be left associative. We define n-ary products $(X_1 \times \dots \times X_n)$ as nested binary products $((\dots (X_1 \times X_2) \times \dots) \times X_n)$. Accordingly we define n-ary tuples $\langle x_1, \dots, x_n \rangle$ as nested binary pairs $\langle \dots \langle x_1, x_2 \rangle, \dots \rangle, x_n \rangle$. Furthermore we define

Expressions $E ::=$	$E * E$	{sequential composition}
	E / E	{parallel composition}
	$E ? E$	{non-deterministic choice}
	id	{identity}
	fork	{sharing}
	unit	{discarding}
	unknown	{free variable}
	fst	{select first term in tuple}
	snd	{select second term in tuple}
	s	$\{s \in \Sigma, \text{ operator or constructor}\}$
	invc	$\{c \in \text{cons}(\Sigma), \text{ inverted constructor}\}$

Fig. 1. Point-Free Expressions

$\langle \dots \langle x_1, x_2 \rangle, \dots \rangle, x_n \rangle$ to be $\langle \rangle$ if $n = 0$ and accordingly $(X_1 \times \dots \times X_n)$ to be $\mathbf{1}$ if $n = 0$, where $\mathbf{1} = \{\langle \rangle\}$. Relations $v : \mathbf{1} \leftrightarrow X$ are called *vectors*.

Instead of using binary direct sums we employ a generalized version that can be defined by means of the injections of binary sums ι_1 and ι_2 . A generalized injection $\iota_{n,k}$ injects a value to the k -th position of an n -ary sum. An n -ary sum is represented by a right parenthesised binary sum. Details on relation-algebraic domain constructions can be found in [14, 17].

2 The Relation Algebraic Semantics

A considerable step towards a relation algebraic semantics has been taken in [5]. There we have presented a transformation from arbitrary functional logic programs to a point-free subset of the same language. The resulting point-free programs are based on a small set of point-wise defined primitives. The term “point-wise” describes that these primitives explicitly access argument variables. The “point-free” declarations are composed of these primitives and do not access their argument variables. In this section we first describe the point-free subset of Curry and the transformation from arbitrary Curry programs into this subset. Then we give a relation algebraic interpretation of the point-wise primitives and the point-free programs based on these primitives.

2.1 Point-Free Curry Programs

Definition 6 presents the syntax of programs that are yielded by the transformation proposed in [5]. Definition 7 presents the declarations of the point-wise primitives which the point-free programs are based on.

Definition 6 (Point-Free Programs). *Let Σ be a constructor-based signature. Then a point-free program over Σ associates each symbol $f \in \text{op}(\Sigma)$ with an expression E of the form defined in Figure 7.*

It is beyond the scope of this paper to give a complete formal definition of the transformation from arbitrary Curry programs to the point-free subset. Rather,

we sketch the key ideas, give some examples and refer the interested reader to [5]. In the resulting program all constructors take exactly one argument. All constant constructors, i.e., those without arguments like `True`, of some type τ are replaced by constructor symbols of type $() \rightarrow \tau$. For example, the definition of `Bool` from Example 1 now reads `data Bool = True () | False ()`. Furthermore, all declarations with more than one argument take a nested structure of binary tuples. This way all arguments can be accessed by the selectors `fst` and `snd`. For example, the definition of `List a` becomes `data List a = Nil () | Cons (a,List a)` and the type of `iff` becomes `iff :: (Bool,Bool) -> Bool`. For all constructors an *inverted* constructor (also called destructor) is added which is defined point-wise and is used to perform pattern matching. For Example, the program from Example 1 is extended by the declarations `invTrue (True x) = x` and `invCons (Cons x) = x`. The following definition provides the declarations of the primitives.

Definition 7 (Point-Wise Primitives). *Point-free programs are based on the following point-wise primitives.*

$(*) :: (a \rightarrow b) \rightarrow (b \rightarrow c) \rightarrow a \rightarrow c$	<code>id :: a -> a</code>
$(f * g) x = g (f x)$	<code>id x = x</code>
$(/) :: (a \rightarrow c) \rightarrow (b \rightarrow d) \rightarrow (a,b) \rightarrow (c,d)$	
$(f / g) (x,y) = (f x, g y)$	
$(?) :: (a \rightarrow b) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow b$	<code>fork :: a -> (a,a)</code>
$(f ? g) x = f x$	<code>fork x = (x,x)</code>
$(f ? g) x = g x$	
<code>unknown :: () -> a</code>	<code>unit :: a -> ()</code>
<code>unknown () = let x free in x</code>	<code>unit x = ()</code>
<code>fst :: (a,b) -> a</code>	<code>snd :: (a,b) -> b</code>
<code>fst (x,y) = x</code>	<code>snd (x,y) = y</code>

Using inverted constructors and the primitives, the definition of `head`, for example, is translated to `head = invCons * fst`. Variables are replaced by `id` where necessary and complex expressions of the form $(s \ t_1 \dots t_n)$ are transformed to $((t'_1 / \dots / t'_n) * s)$ where t'_i are the transformed sub-expressions. For example, $(\text{iff} \ (\text{not } x) \ y)$ becomes $(\text{not}/\text{id}) * \text{iff}$. Sharing of variables is induced by `fork`. For example, $(\text{iff } x \ x)$ becomes $(\text{fork} * \text{iff})$. Free variables are introduced by `unknown`, e.g., $(\text{let } x \text{ free in not } x)$ is transformed to $(\text{unknown} * \text{not})$. Discarded arguments require the introduction of `unit`, for example, the declaration `f x = True` becomes `f = unit * True`. The transformed rules of a function declaration are composed by the non-deterministic choice operator $(?)$ and an additional choice `unit * U` is added. This `U` is the new polymorphic constructor described in Section 1.1 and the additional choice has the effect that any unevaluated expression can be replaced by `U` at any time. This directly corresponds to the Discard Step which has been described in Section 1.1, Definition 5.

Example 11 (Transforming a Complete Function Declaration). The declaration

```
isNil :: List a -> Bool
isNil Nil      = True
isNil (Cons x y) = False
```

is transformed to

```
isNil :: List a -> Bool
isNil = (invNil * True) ? (invCons * unit * False) ? (unit * U)
```

The choice $(\text{unit} * U)$ is added to the transformed version of each function declaration of the original program. This has the same effect as adding a rule “ $f \ x = U$ ” for each user defined operation symbol f . The additional choice makes it possible to evaluate the resulting program with a *strict* semantics and still obtain equivalent results in comparison to the original *lazy* program. An according proof is contained in [5].

A key point of the transformation is that values become mappings from $\text{unit} ()$ to the original value, i.e., *values become vectors*. For example, we transform the expression $(\text{not } \text{True})$, which evaluates to the value **False**, to $(\text{True} * \text{not})$. This expression defines the mapping $\{() \mapsto \text{False}\}$. Therefore, evaluating the expression $(\text{True} * \text{not}) ()$ in the transformed program yields **False**.

In the following, we present a semantics that maps point-free programs to a set of relation algebraic equations. The semantics of an operator models the input/output relation of the declared operation.

2.2 Values, Constructors and Destructors

First we define the sets of values the semantics is based on. The lazy setting requires to introduce partial values. As described in Section 1.1, all values are constructor terms. Partial values contain the special constructor U . Thus, the *set of partial values* is $PV := T_{\text{cons}(\Sigma) \cup \{U\}}(\mathcal{X})$.

In order to model the construction of values we make use of the relation algebraic concept of generalized *direct sums* and their associated *injection* $\iota_{n,k}$ as well as *direct products* and their associated *projections* π, ρ .

Let $c^n \in \text{cons}(\Sigma) \cup \{U\}$ and no be an enumeration of the elements of $\text{cons}(\Sigma) \cup \{U\}$, i.e., a bijective mapping from $\text{cons}(\Sigma) \cup \{U\}$ to $\{1, \dots, |\text{cons}(\Sigma) \cup \{U\}|\}$. Instead of stating k and n explicitly we use injections of the form $\text{inj}_c = \iota_{k,n}$ where $k = no(c)$ and $n = |\text{cons}(\Sigma) \cup \{U\}|$.

Definition 8 (Semantics of Constructors and Destructors)

The semantics of $c \in \text{cons}(\Sigma)$ is defined on base of the injection inj_c by:

$$\llbracket c \rrbracket := \text{inj}_c$$

Furthermore, the destructor corresponding to c is defined as $\llbracket \text{inv}c \rrbracket = \llbracket c \rrbracket^T$.

In the model of the concrete relation algebra the semantics of c has the type $\underbrace{PV \times \dots \times PV}_n \leftrightarrow PV$ and is given by the following set.

$$\llbracket c \rrbracket = \{ \langle \langle x_1, \dots, x_n \rangle, c \ x_1 \ \dots \ x_n \rangle \mid x_1, \dots, x_n \in PV \}$$

Example 12 (Value Semantics). According to Definition 8 the semantics of **Cons** and **Nil** of the signature of Boolean lists, cf. Example 1, are defined by:

Constructor	Abstract Model	Concrete Model
Cons	$inj_{\mathbf{Cons}}$	$\{\langle\langle x, y \rangle, \mathbf{Cons} \ x \ y \rangle \mid x, y \in PV\}$
Nil	$inj_{\mathbf{Nil}}$	$\{\langle\langle \rangle, \mathbf{Nil} \rangle\}$

Definition 9 (Semantics of Declared Operations)

Each operator symbol $f^n \in ops(\Sigma)$ is mapped to a unique variable which ranges over the relations of the appropriate type. Syntactically, we reuse the same symbol and write $\llbracket f \rrbracket := f$.

Note that by Definition 1 $ops(\Sigma) \cap cons(\Sigma) = \emptyset$. The assignment of the variables introduced for $ops(\Sigma)$ is given by the smallest solution of the equation system for the whole program, as given in Definition 15 below.

2.3 Identity, Sequential Composition and Non-deterministic Choice

The primitives for identity **id**, sequential composition $(*)$, and non-deterministic choice $(?)$ have a straight forward correspondence in relation algebra.

Definition 10 (Semantics of **id, $(*)$, $(?)$).** Let e_1, e_2 be point-free expressions as introduced in Definition 7. Then

$$\llbracket \mathbf{id} \rrbracket := \mathbf{I} \quad \llbracket e_1 * e_2 \rrbracket := \llbracket e_1 \rrbracket \circ \llbracket e_2 \rrbracket \quad \llbracket e_1 ? e_2 \rrbracket := \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket$$

Due to Curry being a statically typed language, the type of **I** is never ambiguous.

The next example presents a Curry function and its point-free definition by means of constructors, destructors, $(?)$ and $(*)$.

Example 13 (Semantics of Values and Pattern Matching). Reconsider the definition of Boolean negation in Example 2. Desisting from the details of laziness for the moment, the definition of **not** is transformed to:

```
not :: Bool -> Bool
not = (invTrue * False) ? (invFalse * True)
```

In direct correspondence, the relation algebraic definition is:

$$not = inj_{\mathbf{True}}^T \circ inj_{\mathbf{False}} \cup inj_{\mathbf{False}}^T \circ inj_{\mathbf{True}}$$

As we have illustrated in Section 2.1, pattern matching is defined by a multiplication from the left with the inverse of the constructor semantics. The following lemma justifies this definition, stating that pattern matching with a pattern that corresponds to the outermost constructor peels off the constructor while pattern matching with all other patterns fails.

Lemma 1 (Pattern Matching). Let $c, d \in cons(\Sigma)$. Then we have:

1. $\llbracket c \rrbracket \circ \llbracket c \rrbracket^T = \mathbf{I}$
2. $c \neq d \Rightarrow \llbracket c \rrbracket \circ \llbracket d \rrbracket^T = \mathbf{O}$

Proof. Induction over the structure of inj_c and the basic properties of injections.

Example 14 (Semantics of Pattern Matching). Reconsider the definition of *not* from Example 13. For the application of *not* to the value *True* we get:

$$\begin{aligned} \llbracket \text{True} \rrbracket \circ \text{not} &= \llbracket \text{True} \rrbracket \circ (\llbracket \text{True} \rrbracket^T \circ \llbracket \text{False} \rrbracket \cup \llbracket \text{False} \rrbracket^T \circ \llbracket \text{True} \rrbracket) \\ &= \llbracket \text{True} \rrbracket \circ \llbracket \text{True} \rrbracket^T \circ \llbracket \text{False} \rrbracket \cup \llbracket \text{True} \rrbracket \circ \llbracket \text{False} \rrbracket^T \circ \llbracket \text{True} \rrbracket \\ &= I \circ \llbracket \text{False} \rrbracket \cup O \circ \llbracket \text{True} \rrbracket = \llbracket \text{False} \rrbracket \end{aligned}$$

2.4 Multiple Arguments

The parallel composition operator ($/$) and the tuple selectors *fst* and *snd* are represented using *direct products* and the corresponding *projections*.

Definition 11 (Semantics of $/$, *fst* and *snd*). Let e_1, e_2 be point-free expressions as defined in Definition 6. Then

$$\llbracket e_1 / e_2 \rrbracket := \llbracket e_1 \rrbracket \parallel \llbracket e_2 \rrbracket \quad \llbracket \text{fst} \rrbracket := \pi \quad \llbracket \text{snd} \rrbracket := \rho$$

The type system of Curry ensures that π and ρ are always applied on products of unambiguous type for every appearance in a point-free program.

2.5 Sharing and Call-Time-Choice

In Section 1.1, Example 9, we emphasized that our semantics has to model call-time choice correctly. This means in essence, that shared expressions share non-deterministic choices. In the point-free programs, all sharing is introduced by the primitive *fork*, which is defined employing *tupling*.

Definition 12 (Semantics of *fork*). $\llbracket \text{fork} \rrbracket := [I, I]$

As noted in connection with Definition 11 due to Curry being a statically typed language, the type of I in $[I, I]$ is never ambiguous.

The reason why the presented definition correctly reflects call-time choice can be subsumed as follows. The semantics would be run-time choice iff for any expression e the two applications $\llbracket e \rrbracket * \llbracket \text{fork} \rrbracket$ and $\llbracket \text{fork} \rrbracket * (\llbracket e \rrbracket / \llbracket e \rrbracket)$ are equal. In contrast, in relation algebra the following two properties hold.

Lemma 2. $R \circ [I, I] \subseteq [I, I] \circ (R \parallel R)$
 $R \text{ univalent} \iff R \circ [I, I] = [I, I] \circ (R \parallel R)$

Proof. The first property and $\Rightarrow$ of the second property are implied by the distributivity of $\circ$ over $\cap$. Thus, we only need to show $\Leftarrow$ for the second property:

$$\begin{aligned} R \circ [I, I] = R \circ [I, I] &\iff [R, R] = R \circ [I, I] \\ &\implies [R, R] \circ [I, \bar{I}]^T \subseteq R \circ [I, I] \circ [I, \bar{I}]^T \\ &\iff R \circ I \cap R \circ \bar{I} \subseteq R \circ (I \cap \bar{I}) \\ &\iff R \cap R \circ \bar{I} \subseteq O \\ &\iff R \circ \bar{I} \subseteq \bar{R} \\ &\iff R \text{ univalent (by definition)} \end{aligned}$$

A similar proof is contained in [7, Theorem 4.2].

Example 15 (Call-Time Choice Revisited). Reconsider Example 9. Still desisting from laziness, the point-free versions of `coin` and `iff` are:

```
coin = true ? false
iff  = (invTrue * snd) ? (invFalse * snd * not)
```

By the previous definitions, in the concrete model, *iff* and *coin* are assigned with the following sets:

$$\begin{aligned} iff &= \{ \langle \langle \text{True}, \text{True} \rangle, \text{True} \rangle, \langle \langle \text{True}, \text{False} \rangle, \text{False} \rangle, \\ &\quad \langle \langle \text{False}, \text{False} \rangle, \text{True} \rangle, \langle \langle \text{False}, \text{True} \rangle, \text{False} \rangle \} \\ coin &= \{ \langle \langle \rangle, \text{True} \rangle, \langle \langle \rangle, \text{False} \rangle \} \end{aligned}$$

As explained in Example 9, the expression *shared* := `coin * fork * iff` has a different semantics than *indep* := `fork * (coin/coin) * iff`, the first being a shared call to `coin` whereas the second contains two independent calls to `coin`.

$$\begin{aligned} \llbracket shared \rrbracket &= coin \circ [I, I] \circ iff \\ \llbracket indep \rrbracket &= [I, I] \circ (coin \parallel coin) \circ iff = [coin, coin] \circ iff \end{aligned}$$

By definition of tupling $coin \circ [I, I] = \{ \langle \langle \rangle, \langle \text{True}, \text{True} \rangle \rangle, \langle \langle \rangle, \langle \text{False}, \text{False} \rangle \rangle \}$ whereas $[coin, coin]$ associates all possible pairs over the set $\{\text{True}, \text{False}\}$ with $\langle \rangle$. Therefore, we get, as intended:

$$\llbracket shared \rrbracket = \{ \langle \langle \rangle, \text{True} \rangle \} \quad \llbracket indep \rrbracket = \{ \langle \langle \rangle, \text{True} \rangle, \langle \langle \rangle, \text{False} \rangle \}$$

2.6 Laziness and Demand

In Section 1.1, Example 8, we have seen that lazy functional logic languages allow the declaration of potentially infinite data structures like `true`s. To model laziness we have already introduced the polymorphic constructor U , which is represented in relation algebra as injection, like any other constructor, cf. Section 8. In addition to this constructor we also need to represent the primitive `unit :: a -> ()`, which allows to discard an arbitrary expression without evaluating it. Along with `unit` we define the relation U as a useful abbreviation.

Definition 13 (Semantics of `unit` and Relation U). $\llbracket \text{unit} \rrbracket := \mathsf{L}$ and $\mathsf{U} := \mathsf{L} \circ \text{inj}_{\mathsf{U}}$.

The semantics of `unit` and the relation U inherit well defined types from the types of the Curry program.

Lemma 3 (Laziness). *For all relations R and $c \in \text{cons}(\Sigma)$ it holds that:*

1. $(R \cup \mathsf{U}) \circ \llbracket \text{unit} \rrbracket = \llbracket \text{unit} \rrbracket$ and $(R \cup \mathsf{U}) \circ \mathsf{U} = \mathsf{U}$
2. $[Q, R \cup \mathsf{U}] \circ \llbracket \text{fst} \rrbracket = Q$
3. $[R \cup \mathsf{U}, Q] \circ \llbracket \text{snd} \rrbracket = Q$
4. $(R \cup \mathsf{U}) \circ \llbracket c \rrbracket^{\mathsf{T}} = R \circ \llbracket c \rrbracket^{\mathsf{T}}$

Proof. 1. $(R \cup U) \circ \llbracket \text{unit} \rrbracket \{ \text{def} \} = (R \cup U) \circ L \circ \text{inj}_U$
 $\{ \text{inj}_U \text{ total} \Rightarrow R \cup U \text{ total} \} = L \circ \text{inj}_U \{ \text{def} \} = \llbracket \text{unit} \rrbracket$

2. By Definition 7 we have $\llbracket \text{fst} \rrbracket = \pi$ and we can use [15, Proposition 4.2.2.iii] which states $R \text{ univalent} \Rightarrow (Q \cap S \circ R^T) \circ R = Q \circ R \cap S$ to get:

$$\begin{aligned} [Q, R \cup U] \circ \llbracket \text{fst} \rrbracket &= (Q \circ \pi^T \cap (R \cup U) \circ \rho^T) \circ \pi \\ \{ \pi \text{ univalent} \} &= (Q \circ \pi^T \circ \pi \cap (R \cup U) \circ \rho^T \circ \pi) \\ \{ \text{properties of } \cdot^T, \pi, \rho \} &= Q \cap (R \cup U) \circ L \\ \{ (R \cup U) \text{ total} \} &= Q \cap L = Q \end{aligned}$$

The proof of 3. is analogous to that of 2.

4. The claim stems directly from the properties of injection.

Combining the simple relations U and $\llbracket \text{unit} \rrbracket$ in the way described in Lemma 3 is the center piece of our approach to model laziness. In a lazy framework the value of an expression is either demanded or not demanded. Not being demanded means that the expression is discarded by an application of one of the operations **unit**, **fst** or **snd**. Let R be the semantics $\llbracket e \rrbracket$ of some expression e . Then by adding the relation U , yielding $R \cup U$, we make sure that each expression is indeed “discardable”, i.e., the result of applying **unit**, **fst** or **snd** in an appropriate situation does not depend on R . This is the intention of Lemma 3, 1.-3. The fourth proposition of Lemma 3 covers the case that the value of an expression e is demanded. Demand in a lazy functional logic language is always induced by *pattern matching*, which means in the relation algebraic representation an application of a destructor. If a destructor is applied, the result does only depend on R , while the relation U does not have any impact.

Example 16 (Laziness). Reconsider the declarations of **head** and **true**s from Examples 2 and 8. In the next subsection we define the relation algebraic semantics of these declarations to be the smallest fixpoint of the following equations.

$$\begin{aligned} \text{trues} &= [I, I] \circ (\llbracket \text{True} \rrbracket \parallel \text{trues}) \circ \llbracket \text{Cons} \rrbracket \cup U \\ \text{head} &= \llbracket \text{Cons} \rrbracket^T \circ \pi \cup U \end{aligned}$$

For the application $\text{trues} \circ \text{head}$ we get:

$$\begin{aligned} \text{trues} \circ \text{head} &= ([I, I] \circ (\llbracket \text{True} \rrbracket \parallel \text{trues}) \circ \llbracket \text{Cons} \rrbracket \cup U) \circ (\llbracket \text{Cons} \rrbracket^T \circ \pi \cup U) \\ &= ([\llbracket \text{True} \rrbracket, \text{trues}] \circ \llbracket \text{Cons} \rrbracket \cup U) \circ \llbracket \text{Cons} \rrbracket^T \circ \pi \\ &\quad \cup ([\llbracket \text{True} \rrbracket, \text{trues}] \circ \llbracket \text{Cons} \rrbracket \cup U) \circ U \\ \{\text{Lem 3(1.)}\} &= ([\llbracket \text{True} \rrbracket, \text{trues}] \circ \llbracket \text{Cons} \rrbracket \cup U) \circ \llbracket \text{Cons} \rrbracket^T \circ \pi \cup U \\ \{\text{Lem 3(4.)}\} &= [\llbracket \text{True} \rrbracket, \text{trues}] \circ \llbracket \text{Cons} \rrbracket \circ \llbracket \text{Cons} \rrbracket^T \circ \pi \cup U \\ \{\text{Lem 1}\} &= [\llbracket \text{True} \rrbracket, \text{trues}] \circ I \circ \pi \cup U \\ \{\text{Lem 3(2.)}\} &= \llbracket \text{True} \rrbracket \cup U \end{aligned}$$

2.7 Free Variables

Curry allows declarations of the form **let** x **free** **in** e , where e is an expression. The intended meaning is that free variables are substituted with constructor

terms as needed to compute the normal form of a given expression, cf, Section 1.1. The transformation employs the operation `unknown` to introduce free variables.

Definition 14 (Semantics of `unknown`). $\llbracket \text{unknown} \rrbracket := L$.

The unambiguity of the type of L in each context is ensured by Curry's type system.

By definition, the range of $\llbracket \text{unknown} \rrbracket$ is the set of all partial values. This indeed captures the intended semantics of free variables, because the partial values model the case that a variable has been substituted with a term containing other variables. The notion of an identity on free variables needed in other frameworks is not necessary here. A variable can only appear at different positions of a constructor term if it was shared. Therefore, the call-time choice mechanism considered in the previous section correctly takes care of this case.

Example 17 (Free Variables). Applying the function `not` from Example 13 to a free variable, i.e., evaluating `(let x free in not x)`, yields non-deterministically `True` or `False`, as does the result of its transformation `unknown * not`. The semantics associated with `not` is:

$$\text{not} = \{\langle \text{True}, \text{False} \rangle, \langle \text{False}, \text{True} \rangle\} \cup U$$

Evaluating `unknown * not` in the context of this program yields, as intended:

$$\llbracket \text{unknown} \rrbracket \circ \text{not} = L \circ \text{not} = \{\langle \langle \rangle, \text{False} \rangle, \langle \langle \rangle, \text{True} \rangle\} \cup U$$

Likewise, sharing the free variable, e.g., `(let x free in iff (not x) x)` yields `False` as does the transformed expression `unknown * fork * (not/id) * iff`. Accordingly, the associated relation algebraic expression yields the intended semantics for the same reasons discussed above in Example 15.

$$\begin{aligned} \llbracket \text{unknown} \rrbracket \circ \llbracket \text{fork} \rrbracket \circ (\text{not} \parallel \llbracket \text{id} \rrbracket) \circ \text{iff} &= L \circ [I, I] \circ (\text{not} \parallel I) \circ \text{iff} \\ &= \{\langle \langle \rangle, \text{False} \rangle\} \cup U \end{aligned}$$

2.8 Programs

The last missing step is associating a complete program $\mathcal{P}$ with a semantics. This is done by constructing a relation algebraic equation system from the declarations in $\mathcal{P}$. A solution of the resulting equation system provides the relations to be assigned to the variables which correspond to the user defined operation symbols $f \in \text{ops}(\Sigma)$, cf. Definition 9.

For the according definition recall from Section 2.1 that each declaration for an operator symbol f in a point-free program is of the form $f = e$, where e is an expression according to Definition 6. Therefore, a point-free program is a mapping of the elements of $\text{ops}(\Sigma)$ to the set of point-free expressions.

Definition 15 (Semantics of Programs). Let $\mathcal{P}$ be a point-free program. The semantics of $\mathcal{P}$ is the smallest solution of the set of equations $\{f = \llbracket e \rrbracket \mid f = e \in \mathcal{P}\}$.

Since we do not use any form of relation algebraic negation we do only consider fixpoints of monotonic functionals. Therefore the fixpoint theorem by Tarski can be applied and guarantees the existence of the fixpoints required in Definition 15.

Example 18 (Program Semantics). Recall the declarations and equations of **head** and **true**s in the Examples 8 and 16. In the concrete model the semantics of the program is:

$$\begin{aligned} \text{true}s &= \{ \langle \langle \rangle, \mathbf{U} \rangle, \langle \langle \rangle, \mathbf{True}:\mathbf{U} \rangle, \langle \langle \rangle, \mathbf{True}:\mathbf{True}:\mathbf{U} \rangle \dots \} \\ \text{head} &= \{ \langle \mathbf{Cons} \ x \ y, x \rangle \mid x, y \in PV \} \cup \{ \langle z, \mathbf{U} \rangle \mid z \in PV \} \end{aligned}$$

The semantics associated with *true*s is identical to the standard approaches to model laziness, which employ ideals in complete partial orders (CPO) for functional programming or cones for functional logic programs respectively, cf. 9. We think that the beauty of the presented approach is that no additional concepts like a CPO are needed when using relation algebra. In this way a uniform and high level framework is available for semantics which could be extended for program analysis, partial evaluation, etc. without further additions.

3 Related and Future Work

There are several semantics for functional logic languages, capturing various levels of abstraction. The most abstract approach was first presented in 9 and has been extended in several subsequent works. The introduction in Section 1.1 is essentially a variant of the semantics of 9. One of the main motivation for the approaches following 9, e.g., 11 based on a Launchbury style semantics, 3 based on graph rewriting, and 12 based on rewriting terms with a special **let** construct, was that 9 does not feature an explicit modeling of sharing. The exact operational treatment of sharing, however, frequently proves to be at the cause of semantical difficulties, as worked out, e.g., in 3. All of the above approaches suffer from many technical issues like renaming of variables and various operational details and proofs in the according frameworks often obscure the relevant key ideas. In contrast, we believe that the approach presented in this work provides a framework which is both highly abstract, enabling concise proofs without misleading technical details, while at the same time providing an explicit modeling of sharing.

On the other hand, this work is related to other approaches to capture the semantics of programming languages employing relation algebra. In 4 a relation algebraic semantics for a strict functional programming language is given. In addition to describe *lazy* functional *logic* languages, the presented work also covers algebraic data types and pattern matching where 4 is restricted to Boolean values and if-then-else. Abstract data types are also covered in 16 which provides a relation algebraic framework for lazy functional languages. In comparison to 16, our approach to capture laziness is simpler, not requiring the construction of power sets to remodel the properties of complete partial orders, cf. 16, 6.3]. However, 16 also treats higher order operations, a topic that we have left for future work.

There are several topics for future work. A first one is to prove the equivalence of the presented relation algebraic semantics with the semantics presented in [9]. A second topic concerns the extension of the framework to cover higher order and constraints like term unification. These extensions are usual features of functional logic languages. A third topic is the application of the presented framework to clarify notions diversely discussed in the field of functional logic programming languages, e.g., constructive negation [13], function inversion [2], encapsulated search [3] and sharing of deterministic sub-computations between non-deterministic alternatives [6].

References

1. Albert, E., Hanus, M., Huch, F., Oliver, J., Vidal, G.: Operational semantics for declarative multi-paradigm languages. *Journal of Symbolic Computation* 40(1), 795–829 (2005)
2. Antoy, S., Hanus, M.: Declarative programming with function patterns. In: Hill, P.M. (ed.) *LOPSTR 2005*. LNCS, vol. 3901, pp. 6–22. Springer, Heidelberg (2006)
3. Antoy, S., Braßel, B.: Computing with subspaces. In: Podelski, A. (ed.) *Proceedings of the 9th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pp. 121–30 (2007)
4. Berghammer, R., von Karger, B.: Relational semantics of functional programs. In: *Relational Methods in Computer Science, Advances in Computing Science*, pp. 115–130. Springer, Heidelberg (1997)
5. Braßel, B., Christiansen, J.: Denotation by transformation - towards obtaining a denotational semantics by transformation to point-free style. In: King, A. (ed.) *LOPSTR 2007*. LNCS, vol. 4915, Springer, Heidelberg (2008)
6. Braßel, B., Huch, F.: On a tighter integration of functional and logic programming. In: Shao, Z. (ed.) *APLAS 2007*. LNCS, vol. 4807, pp. 122–138. Springer, Heidelberg (2007)
7. Chin, L.H., Tarski, A.: Distributive and modular laws in the arithmetic of relation algebras. *Univ. of California, Publ. of Mathematics* 1, 341–384 (1951)
8. Echahed, R., Janodet, J.-C.: Admissible graph rewriting and narrowing. In: *Proc. Joint International Conference and Symposium on Logic Programming (JICSLP 1998)*, pp. 325–340 (1998)
9. González-Moreno, J.C., Hortalá-González, M.T., López-Fraguas, F.J., Rodríguez-Artalejo, M.: An approach to declarative programming based on a rewriting logic. *J. Log. Program.* 40(1), 47–87 (1999)
10. Hanus, M.: Multi-paradigm declarative languages. In: Dahl, V., Niemelä, I. (eds.) *ICLP 2007*. LNCS, vol. 4670, pp. 45–75. Springer, Heidelberg (2007)
11. Hughes, J.: Why functional programming matters. In: Turner, D.A. (ed.) *Research Topics in Functional Programming*, pp. 17–42. Addison-Wesley, Reading (1990)
12. López-Fraguas, F.J., Rodríguez-Hortalá, J., Sánchez-Hernández, J.: A simple rewrite notion for call-time choice semantics. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP 2007)*, pp. 197–208. ACM Press, New York (2007)
13. López-Fraguas, F.J., Sánchez-Hernández, J.: Narrowing failure in functional logic programming. In: Hu, Z., Rodríguez-Artalejo, M. (eds.) *FLOPS 2002*. LNCS, vol. 2441, pp. 212–227. Springer, Heidelberg (2002)

14. Maddux, R.D.: Relation-algebraic semantics. *Theoretical Computer Science* 160(1–2), 1–85 (1996)
15. Schmidt, G., Ströhlein, T.: Relations and Graphs - Discrete Mathematics for Computer Scientists. In: *EATCS Monographs on Theoretical Computer Science*, Springer, Heidelberg (1993)
16. Zierer, H.: Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung. PhD thesis, Technische Universität München, Fakultät für Informatik (1988)
17. Zierer, H.: Relation algebraic domain constructions. *Theor. Comput. Sci.* 87(1), 163–188 (1991)

Latest News about Demonic Algebra with Domain

Jean-Lou De Carufel and Jules Desharnais

Département d'informatique et de génie logiciel, Pavillon Adrien-Pouliot,
1065, avenue de la Médecine, Université Laval, Québec, QC, Canada G1V 0A6
jldec1@ift.ulaval.ca, Jules.Desharnais@ift.ulaval.ca

Abstract. We first recall the concept of Kleene algebra with domain (KAD) and how demonic operators can be defined in this algebra. We then present a new axiomatisation of demonic algebra with domain (DAD). It has fewer axioms than the one given in our RelMiCS 9 paper and the axioms are introduced in a way that facilitates comparisons with KAD. The goal in defining DAD is to capture the essence of the demonic operators as defined in KAD. However, not all DADs are isomorphic to a KAD with demonic operators. We characterise those that are by solving a conjecture stated in the RelMiCS 9 paper. In addition, we present new facts about the independence of the axioms.

1 Introduction

Various algebras for program refinement were invented [10,11,12,18,19,20] recently. The demonic refinement algebra (DRA) of von Wright is an abstraction of predicate transformers, while the laws of programming of Hoare et al. have an underlying relational model. Möller's lazy Kleene algebra has weaker axioms than von Wright's and can handle systems in which infinite sequences of states may occur.

This paper goes along similar lines of thought by proposing an abstract algebra for program refinement called *demonic algebra with domain* (DAD). At first, when we defined DAD (see [34]), our goal was to get as close as possible to the kind of algebras that one gets by defining demonic operators in Kleene algebra with domain (KAD), as is done in [8,9], and then forgetting the basic angelic operators of KAD. We called the structure obtained that way demonic algebra with domain (DAD). Then we asked whether or not every DAD is isomorphic to a KAD-based DAD. This is a continuation of the work presented in [34], where it was already shown that DADs and KAD-based DADs are not isomorphic¹. Our contributions in this paper consist mainly of the following:

1. A new axiomatisation of demonic algebra with domain (DAD). It has fewer axioms than the one given in [34] and the axioms are introduced in a way that facilitates comparisons with KAD.

¹ Space constraints force us to tersely recall the basics of demonic algebra. We suggest reading [4] for details.

2. We characterise those DADs which are isomorphic to KAD-based DADs.
3. We present new facts about the independence of the axioms.

In Sect. 2 we recall the definitions of Kleene algebra and its extensions, *Kleene algebra with tests* (KAT) and *Kleene algebra with domain* (KAD). This section also contains the definitions of the demonic operators in terms of the KAD operators. Section 3 presents the concepts of *demonic algebra* (DA) and its extensions, *DA with tests* (DAT), *DA with domain* (DAD) and *DAD with $\neg$* (DAD- $\neg$) as well as derived laws. The definitions presented there are more in line with the standard axiomatisation of KAT and KAD than the ones proposed in [34]. In Sect. 4, angelic operators are defined for those DADs that have the property of consisting of *decomposable* elements. These definitions are the same as in [34]. In Sect. 5, we recall the conjecture of [34] and solve it.

2 Kleene Algebra with Domain and KAD-Based Demonic Operators

In this section, we recall basic definitions about KA and its extensions, KAT and KAD. Then we present the KAD-based definition of the demonic operators.

Definition 1 (Kleene algebra). A Kleene algebra (KA) [2,14] is a structure $(K, +, \cdot, *, 0, 1)$ such that the following properties² hold for all $x, y, z \in K$.

$$(x + y) + z = x + (y + z) \quad (1) \qquad 0 \cdot x = x \cdot 0 = 0 \quad (6)$$

$$x + y = y + x \quad (2) \qquad 1 \cdot x = x \cdot 1 = x \quad (7)$$

$$x + x = x \quad (3) \qquad x \cdot (y + z) = x \cdot y + x \cdot z \quad (8)$$

$$0 + x = x \quad (4) \qquad (x + y) \cdot z = x \cdot z + y \cdot z \quad (9)$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \quad (5) \qquad x^* = x^* \cdot x + 1 \quad (10)$$

Addition induces a partial order $\leq$ such that, for all $x, y \in K$,

$$x \leq y \iff x + y = y \quad (11)$$

Finally, the following properties must be satisfied for all $x, y, z \in K$.

$$x \cdot z + y \leq z \implies x^* \cdot y \leq z \quad (12) \qquad z \cdot x + y \leq z \implies y \cdot x^* \leq z \quad (13)$$

To reason about programs, it is useful to have a concept of condition, or test. It is provided by Kleene algebra with tests.

Definition 2 (Kleene algebra with tests). A KA with tests (KAT) [15] is a structure $(K, \text{test}(K), +, \cdot, *, 0, 1, \neg)$ such that $\text{test}(K) \subseteq \{t \mid t \in K \wedge t \leq 1\}$, $(K, +, \cdot, *, 0, 1)$ is a KA and $(\text{test}(K), +, \cdot, \neg, 0, 1)$ is a Boolean algebra.

² Hollenberg has shown that the dual unfold law $x^* = x \cdot x^* + 1$ is derivable from these axioms [13].

In the sequel, we use the letters s, t, u, v for tests and w, x, y, z for arbitrary elements of K .

Definition 3 (Kleene algebra with domain). A KA with domain (KAD) [6, 7, 9] is a tuple $(K, \text{test}(K), +, \cdot, *, 0, 1, \neg, \lceil \cdot \rceil)$ where $(K, \text{test}(K), +, \cdot, *, 0, 1, \neg)$ is a KAT and, for all $x, y \in K$ and $t \in \text{test}(K)$,

$$x \leq \lceil x \cdot x \rceil, \quad (14)$$

$$\lceil t \cdot x \rceil \leq t, \quad (15)$$

$$\lceil x \cdot \lceil y \rceil \rceil \leq \lceil x \cdot y \rceil. \quad (16)$$

These axioms force the test algebra $\text{test}(K)$ to be the maximal Boolean algebra included in $\{x \mid x \leq 1\}$ (see [7]). Property (16) is called *locality*. We are now ready to introduce the demonic operators. Most proofs can be found in [9].

Definition 4 (Demonic refinement). Let x and y be two elements of a KAD. We say that x refines y , noted $x \sqsubseteq_A y$, when $\lceil y \rceil \leq \lceil x \rceil$ and $\lceil y \cdot x \rceil \leq y$.

The subscript A in $\sqsubseteq_A$ indicates that the demonic refinement is defined with the operators of the angelic world. It is easy to show that $\sqsubseteq_A$ is a partial order.

Proposition 5 (Demonic upper semilattice)

1. The partial order $\sqsubseteq_A$ induces an upper semilattice with demonic join $\sqcup_A$:

$$x \sqsubseteq_A y \iff x \sqcup_A y = y.$$

2. Demonic join satisfies the following two properties:

$$x \sqcup_A y = \lceil x \cdot \lceil y \cdot (x + y) \rceil \quad \text{and} \quad \lceil x \sqcup_A y \rceil = \lceil x \sqcup_A \lceil y \rceil \rceil = \lceil x \cdot \lceil y \rceil \rceil.$$

Definition 6 (Demonic composition). The demonic composition of two elements x and y of a KAD, written $x \sqcup_A y$, is defined by $x \sqcup_A y = \neg \lceil x \cdot \neg \lceil y \rceil \cdot x \cdot y \rceil$.

Definition 7 (Demonic star). Let $x \in K$, where K is a KAD. The unary iteration operator $^{\times_A}$ is defined by $x^{\times_A} = x^* \sqcup_A \lceil x \rceil$.

Based on the partial order $\sqsubseteq_A$, one can focus on tests and calculate the demonic meet of tests.

Definition 8 (Demonic meet of tests). For $s, t \in \text{test}(K)$, define $s \sqcap_A t = s + t$.

For all tests s and t , $s \sqsubseteq_A t \iff t \leq s$. Using Proposition 5, this implies that the operator $\sqcap_A$ is really the demonic meet of tests with respect to $\sqsubseteq_A$.

We now define the t -conditional operator $\sqcap_{A_t}$ that generalises the demonic meet of tests to all elements of a KAD. Since the demonic meet of x and y does not exist in general, $x \sqcap_{A_t} y$ is not the demonic meet of x and y , but rather the demonic meet of $t \sqcup_A x$ and $\neg t \sqcup_A y$.

Definition 9 (*t*-conditional). For each $t \in \text{test}(K)$ and $x, y \in K$, the *t*-conditional is defined by $x \sqcap_{At} y = t \cdot x + \neg t \cdot y$. The family of *t*-conditionals corresponds to a single ternary operator $\sqcap_{A\bullet}$ taking as arguments a test t and two arbitrary elements x and y .

The demonic join operator $\sqcup_A$ is used to give the semantics of demonic non deterministic choices and $\sqcup_A$ is used for sequences. Among the interesting properties of $\sqcup_A$, we cite $t \sqcup_A x = t \cdot x$, which says that composing a test t with an arbitrary element x is the same in the angelic and demonic worlds, and $x \sqcup_A y = x \cdot y$ if $\top y = 1$, which says that if the second element of a composition is total, then again the angelic and demonic compositions coincide. The ternary operator $\sqcap_{A\bullet}$ is similar to the conditional choice operator $\_ \triangleleft \_ \triangleright \_$ of Hoare et al. [10,11]. It corresponds to a guarded choice with disjoint alternatives. The iteration operator $^{\times_A}$ rejects the finite computations that go through a state from which it is possible to reach a state where no computation is defined (e.g., due to blocking, abnormal termination or infinite looping).

As usual, unary operators have the highest precedence, and demonic composition $\sqcup_A$ binds stronger than $\sqcup_A$ and $\sqcap_{A\bullet}$, which have the same precedence.

Theorem 10 (KA-based demonic operators). The structure $(K, \text{test}(K), \sqcup_A, \sqcup_A, ^{\times_A}, 0, 1, \neg, \sqcap_A, \top, \sqcap_{A\bullet})$ is a demonic algebra with domain and $\sqcap_{A\bullet}$ as defined in Sect. 3 (Definitions 11, 12, 15 and 23).

3 Axiomatisation of Demonic Algebra with Domain

The demonic operators introduced at the end of the last section satisfy many properties. We choose some of them to become axioms of a new structure called demonic algebra with domain. For this definition, we follow the same path as for the definition of KAD. That is, we first define demonic algebra, then demonic algebra with tests and, finally, demonic algebra with domain.

3.1 Demonic Algebra

Demonic algebra, like KA, has a sum, a composition and an iteration operator.

Definition 11 (Demonic algebra). A demonic algebra (DA) is a structure $(A_D, \sqcup, \sqcap, \times, \top, 1)$ such that the following properties are satisfied for $x, y, z \in A_D$.

$$x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z \quad (17) \qquad \top \sqcup x = x \sqcup \top = \top \quad (22)$$

$$x \sqcup y = y \sqcup x \quad (18) \qquad 1 \sqcup x = x \sqcup 1 = x \quad (23)$$

$$x \sqcup x = x \quad (19) \qquad x \sqcup (y \sqcup z) = x \sqcup y \sqcup x \sqcup z \quad (24)$$

$$\top \sqcup x = \top \quad (20) \qquad (x \sqcup y) \sqcup z = x \sqcup z \sqcup y \sqcup z \quad (25)$$

$$x \times (y \sqcup z) = (x \times y) \sqcup z \quad (21) \qquad x^\times = x^\times \sqcup x \sqcup 1 \quad (26)$$

There is a partial order $\sqsubseteq$ induced by $\sqcup$ such that for all $x, y \in A_D$,

$$x \sqsubseteq y \iff x \sqcup y = y. \quad (27)$$

The next two properties are also satisfied for all $x, y, z \in A_{\mathcal{D}}$.

$$x \sqcap z \sqcup y \sqsubseteq z \implies x^{\times} \sqcap y \sqsubseteq z \quad (28) \qquad z \sqcap x \sqcup y \sqsubseteq z \implies y \sqcap x^{\times} \sqsubseteq z \quad (29)$$

When comparing Definitions [11](#) and [11](#), one observes the obvious correspondences $+ \leftrightarrow \sqcup, \cdot \leftrightarrow \sqcap, * \leftrightarrow \times, 0 \leftrightarrow \top, 1 \leftrightarrow 1$. The only difference in the axiomatisation between KA and DA is that 0 is the left and right identity of addition in KA ($+$), while the corresponding element $\top$ is a left and right zero of addition in DA ($\sqcup$). However, this minor difference has a rather important impact. While KAs and DAs are upper semilattices with $+$ as the join operator for KAs and $\sqcup$ for DAs, the element 0 is the bottom of the semilattice for KAs and $\top$ is the top of the semilattice for DAs. Indeed, by [\(22\)](#) and [\(27\)](#), $x \sqsubseteq \top$ for all $x \in A_{\mathcal{D}}$.

All operators are isotone with respect to the refinement ordering $\sqsubseteq$:

$$x \sqsubseteq y \implies z \sqcup x \sqsubseteq z \sqcup y \wedge z \sqcap x \sqsubseteq z \sqcap y \wedge x \sqcap z \sqsubseteq y \sqcap z \wedge x^{\times} \sqsubseteq y^{\times}.$$

This can easily be derived from [\(18\)](#), [\(19\)](#), [\(23\)](#), [\(24\)](#), [\(25\)](#), [\(26\)](#), [\(27\)](#) and [\(28\)](#).

3.2 Demonic Algebra with Tests

Now comes the first extension of DA, demonic algebra with tests. This extension has a concept of tests like the one in KAT and it also adds the operator $\sqcap$. Introducing $\sqcap$ provides a way to express the meet of tests, as will be shown below. In KAT, $+$ and $\cdot$ are respectively the join and meet operators of the Boolean lattice of tests. But in Sect. [3.3](#), it will turn out that for any tests s and t , $s \sqcup t = s \sqcap t$, so that $\sqcup$ and $\sqcap$ both act as the join operator on tests (this is also the case for the KAD-based definition of these operators given in Sect. [2](#)).

Definition 12 (Demonic algebra with tests). A demonic algebra with tests (DAT) is a structure $(A_{\mathcal{D}}, B_{\mathcal{D}}, \sqcup, \sqcap, \times, \top, 1, \neg, \sqcap)$ such that $\{1, \top\} \subseteq B_{\mathcal{D}} \subseteq A_{\mathcal{D}}$, $(A_{\mathcal{D}}, \sqcup, \sqcap, \times, \top, 1)$ is a DA and $(B_{\mathcal{D}}, \sqcup, \sqcap, \neg, 1, \top)$ is a Boolean algebra. The elements in $B_{\mathcal{D}}$ are called (demonic) tests. The operator $\sqcap$ stands for the infimum of elements in $B_{\mathcal{D}}$ with respect to $\sqsubseteq$.

Note that 1 and $\top$ are respectively the bottom and the top of the Boolean lattice of tests and that $\sqcap$ and $\neg$ are defined exclusively on $B_{\mathcal{D}}$. In the sequel, we use the letters s, t, u, v for demonic tests and w, x, y, z for arbitrary elements of $A_{\mathcal{D}}$.

This definition gives no indication about the behaviour of $\sqcap$ on tests. Example [13](#) is instructive in this respect. It was constructed by Mace4 [\[17\]](#).

Example 13. For this example $A_{\mathcal{D}} = B_{\mathcal{D}} = \{\top, s, t, 1\}$. The demonic operators are defined by the following tables.

$\sqcup$	$\top$	s	t	1	$\sqcap$	$\top$	s	t	1	$\times$	$\neg$	$\sqcap$	$\top$	s	t	1
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	s	$\top$	$\top$	$\top$	s	$\top$	s	s	s	s	s	1
t	$\top$	$\top$	t	t	t	$\top$	$\top$	$\top$	t	$\top$	t	t	t	t	1	t
1	$\top$	s	t	1	1	$\top$	s	t	1	1	$\top$	1	1	1	1	1

A basic property of DAD (see Sect. 3.3) is that $s \sqcap t = s \sqcup t$ (see Proposition 21.3). It turns out that the present algebra is a DAT where $s \sqcap t = s \sqcup t$ does not hold. Indeed, $s \sqcup s = s \neq \top = s \sqcap s$. Note that $s \sqcap (t \sqcap u) = s \sqcap t \sqcap s \sqcap u$ does not hold either. Indeed, $s \sqcap (s \sqcap t) = s \neq \top = s \sqcap s \sqcap s \sqcap t$.

Definition 12 neither tells if $B_{\mathcal{D}}$ is closed under $\sqcap$. The axioms provided by DAD (see Sect. 3.3) will bring light to that question. Before moving to DAD, we have a lemma about DAT.

Lemma 14. *The following refinements hold for all $s, t \in B_{\mathcal{D}}$ and all $x \in A_{\mathcal{D}}$.*

1. $x \sqsubseteq t \sqcap x$	4. $t \sqcap \neg t = \neg t \sqcap t = \top$
2. $x \sqsubseteq x \sqcap t$	5. $1 \sqsubseteq s \sqcap t$
3. $s \sqcup t \sqsubseteq s \sqcap t$	6. $t \sqcap x \sqsubseteq x \implies \top \sqsubseteq \neg t \sqcap x$

3.3 Demonic Algebra with Domain

The next extension consists in adding a domain operator to DAT. It is denoted by the symbol $\ulcorner$.

Definition 15 (Demonic algebra with domain). A demonic algebra with domain (DAD) is a structure $(A_{\mathcal{D}}, B_{\mathcal{D}}, \sqcup, \sqcap, \times, \top, 1, \neg, \sqcap, \ulcorner)$, where $(A_{\mathcal{D}}, B_{\mathcal{D}}, \sqcup, \sqcap, \times, \top, 1, \neg, \sqcap)$ is a DAT and the demonic domain operator $\ulcorner : A_{\mathcal{D}} \rightarrow B_{\mathcal{D}}$ satisfies the following properties for all $t \in B_{\mathcal{D}}$ and all $x, y \in A_{\mathcal{D}}$.

$$\ulcorner(x \sqcap t) \sqcap x = x \sqcap t \quad (30) \quad \ulcorner(x \sqcup y) = \ulcorner x \sqcup \ulcorner y \quad (32)$$

$$\ulcorner(x \sqcap y) = \ulcorner(x \sqcap \ulcorner y) \quad (31) \quad \ulcorner(x \sqcap t) \sqsubseteq t \implies \ulcorner(x \times \sqcap t) \sqsubseteq t \quad (33)$$

Remark 16. As noted above, the axiomatisation of DA is very similar to that of KA, so one might expect the resemblance to continue between DAD and KAD. This is true of (31), which is locality in a demonic world. But, looking at the angelic version of Definition 15, namely Definition 3, one might expect to find axioms like $\ulcorner x \sqcap x \sqsubseteq x$ and $t \sqsubseteq \ulcorner(t \sqcap x)$, or $t \sqsubseteq \ulcorner x \iff t \sqcap x \sqsubseteq x$. These three properties indeed hold in DAD (see Propositions 21.8 and 21.11 and 3.4). However, (30) cannot be derived from these three properties, even when assuming (31), (32) and (33) (see Example 17). Since (30) holds in KAD-based demonic algebras (see Theorem 10) and because our goal is to come as close as possible to these, we include (30) as an axiom.

Examples 17, 18, 19 and 20 illustrate the independence of Axioms (30), (31), (32) and (33). Examples 17, 18 and 19 were constructed by Mace4 17. Example 20 was not since it is infinite. Note that the tables for $\sqcap$ are not given in either of these examples since they can be derived from those for $\neg$ and $\ulcorner$ by De Morgan.

Example 17. For this example $A_{\mathcal{D}} = \{\top, s, t, 1, a, b\}$ and $B_{\mathcal{D}} = \{\top, s, t, 1\}$. The demonic operators are defined by the following tables.

$\sqcup$	$\top$	s	t	1	a	b
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	a <td>b</td>	b
t	$\top$	$\top$	t	t	$\top$	$\top$
1	$\top$	s	t	1	a <td>b</td>	b
a	$\top$	a	$\top$	a	a	b
b	$\top$	b	$\top$	b	b	b

$\sqcap$	$\top$	s	t	1	a	b
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	a <td>b</td>	b
t	$\top$	$\top$	t	t	$\top$	$\top$
1	$\top$	s	t	1	a <td>b</td>	b
a	$\top$	b	$\top$	a	b	b
b	$\top$	b	$\top$	b	b	b

$\times$
$\top$
s
t
1
a
b

$\neg$
1
s
t
1
$\top$

$\neg$
$\top$
s
t
1
a
b

This algebra is a DAT for which $\neg x \sqcap x \sqsubseteq x$, $t \sqsubseteq \neg(t \sqcap x)$, $t \sqsubseteq \neg x \iff t \sqcap x \sqsubseteq x$, (31), (32) and (33) all hold, but (30) does not. Indeed $\neg(a \sqcap s) \sqcap a = a \neq b = a \sqcap s$.

Then why choose (30) rather than $\neg x \sqcap x \sqsubseteq x$ and $t \sqsubseteq \neg(t \sqcap x)$? The justification is twofold. Firstly, as already mentioned in Remark 16, models that come from KAD satisfy property (30). Secondly, there are strong indications that this law is essential to demonstrate most of the results of Sections 4 and 5.

In KAD, it is not necessary to have an axiom like (32), because additivity of $\neg$ follows from the axioms of KAD (Definition 3) and the laws of KAT. The proof that works for KAD does not work here.

Example 18. For this example $A_{\mathcal{D}} = \{\top, s, t, 1, a\}$ and $B_{\mathcal{D}} = \{\top, s, t, 1\}$. The demonic operators are defined by the following tables.

$\sqcup$	$\top$	s	t	1	a	$\sqcap$	$\top$	s	t	1	a	$\times$	$\neg$	$\neg$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	$\top$	s	$\top$	s	$\top$	s	a	s	s	s
t	$\top$	$\top$	t	t	$\top$	t	$\top$	$\top$	t	t	$\top$	t	s	t
1	$\top$	s	t	1	$\top$	1	$\top$	s	t	1	a	1	1	1
a	$\top$	$\top$	$\top$	$\top$	a	a	$\top$	$\top$	$\top$	$\top$	a	a	$\top$	a

This algebra is a DAT and, in addition, (30), (31) and (33) are satisfied, but (32) is not. Indeed $\neg(1 \sqcup a) = \top \neq s = \neg 1 \sqcup \neg a$.

Example 19. For this example $A_{\mathcal{D}} = \{\top, s, t, 1, a, b, c, d\}$ and $B_{\mathcal{D}} = \{\top, s, t, 1\}$. The demonic operators are defined by the following tables.

$\sqcup$	$\top$	s	t	1	a	b	c	d
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	$\top$	s	$\top$	$\top$
t	$\top$	$\top$	t	t	a	d	c	d
1	$\top$	s	t	1	a	b	c	d
a	$\top$	$\top$	a	a	a	c	c	c
b	$\top$	s	d	b	c	b	c	d
c	$\top$	$\top$	c	c	c	c	c	c
d	$\top$	$\top$	d	d	c	d	c	d

$\sqcap$	$\top$	s	t	1	a	b	c	d
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
s	$\top$	s	$\top$	s	$\top$	s	$\top$	$\top$
t	$\top$	$\top$	t	t	a	d	c	d
1	$\top$	s	t	1	a	b	c	d
a	$\top$	$\top$	a	a	a	$\top$	$\top$	$\top$
b	$\top$	s	$\top$	b	$\top$	b	$\top$	$\top$
c	$\top$	$\top$	$\top$	c	$\top$	$\top$	$\top$	$\top$
d	$\top$	$\top$	$\top$	d	$\top$	d	$\top$	$\top$

$\times$
$\top$
s
t
1
a
b
c
d

$\neg$
1
s
t
1

$\neg$
$\top$
s
t
1

In this DAT, (30), (32) and (33) are satisfied, but (31) is not. Indeed $\neg(a \sqcap b) = \top \neq t = \neg(a \sqcap \neg b)$.

Finally, we add Axiom (33) since it is true in KAD-based demonic algebras (see Theorem 10) and because it cannot be deduced from (30), (31) and (32). Indeed, see Example 20.

Example 20. For this example $A_{\mathcal{D}} = \{E \in \wp(\mathbf{N}) : E \text{ is finite}\}$ and $B_{\mathcal{D}} = \{\{\}, \{0\}\}$. The demonic operators are as follows. (1) Demonic join: $E \sqcup F = E \cup F$ if $E \neq \{\}$ and $F \neq \{\}$, and $E \sqcup \{\} = \{\} \sqcup F = \{\}$. (2) Demonic composition: $E \circ F = \{x \in \mathbf{N} : (\exists e \in E, f \in F : x = e + f)\}$. (3) Demonic star: $E^{\times} = \{\}$ if $E \neq \{0\}$, and $\{0\}^{\times} = \{0\}$. (4) Domain: $\ulcorner E = \{0\}$ if $E \neq \{\}$, and $\ulcorner \{\} = \{\}$. Hence $\{\}$ is the top of the upper semilattice $(A_{\mathcal{D}}, \sqsubseteq)$ and $\{0\}$ is neutral for demonic composition. The operators on demonic tests are trivially defined.

In this DAT, (30), (31) and (32) are satisfied, but (33) is not. Indeed $\ulcorner \{1\} \circ \{0\} \sqsubseteq \{0\} \not\Rightarrow \ulcorner \{1\}^{\times} \circ \{0\} \sqsubseteq \{0\}$.

The axioms of DAD impose important restrictions on demonic tests. These restrictions are actually useful properties and they are presented in the following proposition together with properties of $\ulcorner$ (see 34 for more properties).

Proposition 21. *In a DAD, the demonic domain operator satisfies the following properties. Take $x, y \in A_{\mathcal{D}}$ and $s, t, u \in B_{\mathcal{D}}$.*

1. $\ulcorner t = t$	8. $\ulcorner x \circ x = x$
2. $t \circ t = t$	9. $x \sqsubseteq y \Rightarrow \ulcorner x \sqsubseteq \ulcorner y$
3. $s \sqcup t = s \circ t$	10. $\ulcorner (t \circ x) = t \circ \ulcorner x$
4. $s \circ (t \sqcap u) = s \circ t \sqcap s \circ u$	11. $t \sqsubseteq \ulcorner (t \circ x)$
5. $(s \sqcap t) \circ u = s \circ u \sqcap t \circ u$	12. $\ulcorner (x \circ s) \circ \ulcorner (x \circ t) = \ulcorner (x \circ s \circ t)$
6. $s \circ t = t \circ s$	13. $\neg \ulcorner x \circ x = \top$
7. $x \sqsubseteq t \circ y \iff t \circ x \sqsubseteq t \circ y$	14. $\ulcorner x \sqsubseteq \ulcorner (x \circ y)$

All the above laws except 12 are identical to laws of $\ulcorner$, after compensating for the reverse ordering of the Boolean lattice (on tests, $\sqsubseteq$ corresponds to $\geq$). Proposition 21.3 implies that $B_{\mathcal{D}}$ is closed under $\circ$.

Although Proposition 21.1 is a quite basic property, its proof uses (30). Since that axiom is not as natural as the others, it would be interesting to find a proof that only involves (31) and (32). Furthermore, Proposition 21.1 and (30) are used in the proof of Propositions 21.2, 21.3, 21.4, 21.5, 21.6, 21.7 and 21.8. It turns out that it is not possible. Indeed, see Example 22.

Example 22. Consider Example 13 where we add a domain operator defined by $\ulcorner \top = \ulcorner \bot = \ulcorner \mathbf{t} = \ulcorner \mathbf{f} = \top$.

This algebra is a DAT and, in addition, (31), (32) and (33) are satisfied, but (30) and $\ulcorner t = t$ are not. Indeed $\ulcorner (1 \circ 1) \circ 1 = \top \neq 1 = 1 \circ 1$ and $\ulcorner \mathbf{f} = \top \neq 1$. Note that Propositions 21.2, 21.3, 21.4, 21.5, 21.7 and 21.8 are not satisfied either.

For those who wonder, the major difference between Examples 17 and 22 is that $\ulcorner x \circ x \sqsubseteq x$ is satisfied in the former and not in the latter. In conclusion,

$$\ulcorner x \circ x \sqsubseteq x \wedge t \sqsubseteq \ulcorner (t \circ x) \wedge (31) \wedge (32) \not\Rightarrow (30),$$

$$\begin{aligned}
(30) \wedge (31) \wedge (32) &\Longrightarrow \ulcorner x \sqcap x \sqsubseteq x \wedge t \sqsubseteq \ulcorner(t \sqcap x) \urcorner, \\
(31) \wedge (32) &\not\Longrightarrow \ulcorner t = t \urcorner, \\
\ulcorner x \sqcap x \sqsubseteq x \wedge t \sqsubseteq \ulcorner(t \sqcap x) \urcorner \wedge (31) \wedge (32) &\Longrightarrow \ulcorner t = t \urcorner.
\end{aligned}$$

Despite the fact that Proposition 21 can be proved from $\ulcorner x \sqcap x \sqsubseteq x, t \sqsubseteq \ulcorner(t \sqcap x) \urcorner$, (31) and (32), there are crucial results that cannot be derived and for which (30) is necessary. For instance, the proof of the most important theorem of this paper (Theorem 35, Sect. 5.4) and the proof of the most important theorem of [34] (Theorem 28, Section 5) call for (30) many times.

Since in DAD $s \sqcap t = s \sqcup t$ for all $s, t \in B_D$ (see Proposition 21.3), the Boolean algebra of demonic tests B_D may be viewed as $(B_D, \sqcup, \sqcap, \neg, 1, \top)$ or as $(B_D, \sqcap, \sqcup, \neg, 1, \top)$.

3.4 Demonic Algebra with Domain and $\sqcap_\bullet$

The operator $\sqcap$ defined on B_D ensures that demonic tests form a Boolean algebra. In KA, the addition of an analogous operator is not necessary since $\cdot$ already corresponds to the meet of tests. We wish to have an operator defined on A_D (not only on B_D) and the need to make DAD more expressive leads us to the operator $\sqcap_\bullet$. Indeed, in KA the tests and the domain operator were sufficient to define demonic operators. However, some tools are still missing in DAD in order to retrieve angelic operators (see Sect. 4) and the operator $\sqcap_\bullet$ is one of them.

There are two requirements on $\sqcap_\bullet$. Firstly, it has to respect $\sqcap$ when evaluated on demonic tests. Secondly, it should behave like a choice operator.

Definition 23 (Demonic algebra with domain and $\sqcap_\bullet$). A demonic algebra with domain and $\sqcap_\bullet$ ($DAD\text{-}\sqcap_\bullet$) is a structure $(A_D, B_D, \sqcup, \sqcap, \times, \top, 1, \neg, \sqcap, \ulcorner, \sqcap_\bullet)$, where $(A_D, B_D, \sqcup, \sqcap, \times, \top, 1, \neg, \sqcap, \ulcorner)$ is a DAD and the t -conditional operator $\sqcap_\bullet$ is a ternary operator of type $B_D \times A_D \times A_D \rightarrow A_D$ that can be thought of as a family of binary operators. For each $t \in B_D$, $\sqcap_t$ is an operator of type $A_D \times A_D \rightarrow A_D$, and of type $B_D \times B_D \rightarrow B_D$ if its two arguments belong to B_D . It satisfies the following property for all $t \in B_D$ and all $x, y, z \in A_D$.

$$x \sqcap_t y = z \iff t \sqcap x = t \sqcap z \wedge \neg t \sqcap y = \neg t \sqcap z$$

We now present some properties of $\sqcap_t$ (see [34] for more properties).

Proposition 24. Let A_D be a $DAD\text{-}\sqcap_\bullet$. The following properties are true for all $s, t, u \in B_D$ and all $x, x_1, x_2, y, y_1, y_2, z \in A_D$.

1. $t \sqcap (x \sqcap_t y) = t \sqcap x$
2. $\neg t \sqcap (x \sqcap_t y) = \neg t \sqcap y$
3. $x \sqcap_t y = y \sqcap_{\neg t} x$
4. $(t \sqcap x) \sqcap_t y = x \sqcap_t y$
5. $x \sqcap_t (\neg t \sqcap y) = x \sqcap_t y$
6. $x \sqcap_t \top = t \sqcap x \wedge \top \sqcap_t x = \neg t \sqcap x$
7. $(x \sqcap_t y) \sqcap z = x \sqcap z \sqcap_t y \sqcap z$
8. $s \sqcap (x \sqcap_t y) = s \sqcap x \sqcap_t s \sqcap y$
9. $1 \sqcap_s t = s \sqcap t$
10. $s \sqcap_t u = t \sqcap s \sqcap \neg t \sqcap u$
11. $x \sqcap_t x = x$
12. $x \sqsubseteq y \implies x \sqcap_t z \sqsubseteq y \sqcap_t z$
13. $x \sqsubseteq y \implies z \sqcap_t x \sqsubseteq z \sqcap_t y$
14. $\ulcorner(x \sqcap_t y) \urcorner = \ulcorner x \sqcap_t \urcorner y$

15. $x \sqsubseteq y \iff t \sqcap x \sqsubseteq t \sqcap y \wedge \neg t \sqcap x \sqsubseteq \neg t \sqcap y$
 16. The meet with respect to $\sqsubseteq$ of $t \sqcap x$ and $\neg t \sqcap y$ exists and is equal to $x \sqcap_t y$.

If we draw up what we got, tests have quite similar properties in KAT and DAT. But there are important differences as well. The first one is that $\sqcup$ and $\sqcap$ behave the same way on tests (Proposition 21.3). The second one concerns Law 15 of Proposition 24, which show how a proof of refinement can be done by case analysis by decomposing it with cases t and $\neg t$. The same is true in KAT. However, in KAT, this decomposition can also be done on the right side, since for instance the law $x \leq y \iff x \cdot t \leq y \cdot t \wedge x \cdot \neg t \leq y \cdot \neg t$ holds, while the corresponding law does not hold in DAT. With the t -conditional operator, there is an asymmetry between left and right that can be traced back to Propositions 24.7 and 24.8. In Proposition 24.7, right distributivity holds for arbitrary elements, while left distributivity in Proposition 24.8 holds only for tests.

Propositions 24.12 and 24.13 simply express the isotony of $\sqcap_t$ in its two arguments. On the other hand, $\sqcap_\bullet$ is not isotone with respect to its test argument. Proposition 24.9 establishes the link between $\sqcap_\bullet$ and $\sqcap$ and makes it clear that the former is a generalisation of the latter. This is a generalisation since it has the same behaviour on demonic tests and it still calculates a kind of meet with respect to $\sqsubseteq$ on other elements. Indeed, Proposition 24.16 tells us that $x \sqcap_t y$ is the demonic meet of $t \sqcap x$ and $\neg t \sqcap y$.

To simplify the notation when possible, we will use the abbreviation

$$x \sqcap y = x \sqcap_{\neg x} y . \quad (34)$$

It turns out that it is consistent with demonic meet on demonic tests. Under special conditions, $\sqcap$ has easy to use properties, as shown by the next corollary (see 3.4 for more properties).

Corollary 25. *Let x, y, z be arbitrary elements and s, t be tests of a DAD- $\sqcap_\bullet$.*

1. $s \sqcap t$ as defined by (34) is equal to the meet of s and t in the Boolean lattice of tests defined in Definition 12 (so there is no possible confusion).

2. $\top \sqcap x = x \sqcap \top = x$	7. $(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z)$
3. $t \sqcap (x \sqcap y) = t \sqcap x \sqcap t \sqcap y$	8. $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$
4. $(s \sqcap t) \sqcap x = s \sqcap x \sqcap t \sqcap x$	9. $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$
5. $\neg x \sqcap y = \neg y \sqcap x \implies x \sqcap y = y \sqcap x$	10. $\neg(x \sqcap y) = \neg x \sqcap \neg y$
6. $\neg x \sqcap \neg y = \top \implies \neg x \sqcap y = \neg y \sqcap x$	11. $\neg x \sqcap \neg y = \top \implies (x \sqcap y) \sqcap z = x \sqcap z \sqcap y \sqcap z$

Remark 26. Propositions 24.16 and 21.8 with (34) imply that $x \sqcap y$ is the infimum of x and $\neg \neg x \sqcap y$ with respect to $\sqsubseteq$. Propositions 24.9 and 24.4, (34) and Corollary 25.1 imply that $s \sqcap t$ is well defined as the infimum of s and t in the Boolean lattice of demonic tests $B_{\mathcal{D}}$.

With this new axiomatisation (compared to 3.4), we only add a Boolean algebra to DA to get DAT rather than adding a Boolean algebra together with a $\sqcap_\bullet$ operator that acts on all elements. This is more like for KAT (see 15). Then we

add a domain operator that is almost the same as the one introduced in [34]. It turns out that we nevertheless recover the previous properties of demonic tests and domain. Finally, with all these tools, we only need one law to define the t -conditional operator, which is a worth noting improvement.

4 Definition of Angelic Operators in DAD

In this section, we recall the definition of angelic operators from the demonic ones introduced in [34].

4.1 Angelic Refinement and Angelic Choice

Definition 27 (Angelic refinement and angelic choice). *Let x, y be elements of a $DAD-\mathbb{F}_\bullet$. We say that $x \leq_D y$ when $\ulcorner y \sqsubseteq \urcorner x$ and $x \sqsubseteq \urcorner x \Box y$. We define the operator $+_D$ by $x +_D y = (x \sqcup y) \sqcap \neg \urcorner y \Box x \sqcap \neg \urcorner x \Box y$.*

Proposition 28 (Angelic choice). *In a $DAD-\mathbb{F}_\bullet$, $A_{\mathcal{D}}$, $\leq_D$ is a partial order satisfying $x \leq_D y \iff x +_D y = y$ for all $x, y \in A_{\mathcal{D}}$.*

4.2 Angelic Composition and Demonic Decomposition

We now turn to the definition of angelic composition. But things are not as simple as for $\leq_D$ or $+_D$. The difficulty is due to the asymmetry between left and right caused by the difference between Propositions [24.7] and [24.8], and by the absence of a codomain operator for “testing” the right-hand side of elements as can be done with the domain operator on the left.

In order to circumvent that difficulty, we need the concept of decomposition. See [34] for an intuitive justification of its introduction.

Definition 29. *Let t be a test. An element x of a $DAD-\mathbb{F}_\bullet$ is said to be t -decomposable iff there are unique elements x_t and $x_{\neg t}$ such that*

$$x = x \Box t \sqcap x \Box \neg t \sqcap (x_t \sqcup x_{\neg t}) , \quad (35)$$

$$\urcorner(x_t) = \urcorner(x_{\neg t}) = \neg \urcorner(x \Box t) \Box \neg \urcorner(x \Box \neg t) \Box \urcorner x , \quad (36)$$

$$x_t = x_t \Box t , \quad (37)$$

$$x_{\neg t} = x_{\neg t} \Box \neg t . \quad (38)$$

And x is said to be decomposable iff it is t -decomposable for all tests t .

And then we define angelic composition.

Definition 30 (Angelic composition). *Let x and y be elements of a $DAD-\mathbb{F}_\bullet$ such that x is decomposable. Then the angelic composition $\cdot_{\mathcal{D}}$ is defined by*

$$x \cdot_{\mathcal{D}} y = x \Box y \sqcap x_{\neg y} \Box y .$$

4.3 Kleene Star

Finally, here is the definition of angelic iteration, which is slightly different from the one presented in [34], but more usable that way. Moreover, the two definitions are equivalent.

Definition 31 (Angelic iteration). *Let x be an element of a $\text{DAD-}\mathfrak{F}$. The angelic finite iteration operator ^{*D} is defined by $x^{*D} = (x \sqcap 1)^\times$.*

5 The Conjecture

We begin this section by recalling the conjecture introduced in [34].

Conjecture 32 (Subalgebra of decomposable elements).

1. The set of decomposable elements of a $\text{DAD-}\mathfrak{F}$, $A_{\mathcal{D}}$ is a subalgebra of $A_{\mathcal{D}}$.
2. For the subalgebra of decomposable elements of $A_{\mathcal{D}}$, the composition $\mathcal{D}$ is associative and distributes over $+_D$ (properties (5), (8) and (9)).
3. For the subalgebra of decomposable elements of $A_{\mathcal{D}}$, the iteration operator ^{*D} satisfies the unfolding and induction laws of the Kleene star (properties (10), (12) and (13)).

The following list contains new facts about decomposition and answers to Conjecture 32.

- The demonic tests are decomposable (see [34]).
- There is a $\text{DAD-}\mathfrak{F}$, where some elements are not decomposable (see [34]).
- Let t be a demonic test. An element of a $\text{DAD-}\mathfrak{F}$, may have more than one t -decomposition, in other words, it is relevant to ask for “unicity” in Definition 29 (see Sect. 5.1).
- The first point of Conjecture 32 is false: there is a $\text{DAD-}\mathfrak{F}$, containing decomposable elements a and b such that $a \sqcup b$ is not decomposable (see Sect. 5.2). It turns out that this has only a minor impact on the other parts of the conjecture.
- Therefore we consider maximal subalgebras of decomposable elements that are not necessarily composed of all decomposable elements (see Sect. 5.3).
- In a subalgebra $I \supseteq B_{\mathcal{D}}$ of decomposable elements of a $\text{DAD-}\mathfrak{F}$, $A_{\mathcal{D}}$, $(I, B_{\mathcal{D}}, +_D, \mathcal{D}, ^{*D}, \top, 1, \neg, \lceil \rceil)$ is a KAD (see Sect. 5.4).

5.1 Multiple Decomposition for a Single Element

The following example is one where there are x and t such that the t -decomposition of x is not unique. This example is constructed from the general structure introduced in the following lemma.

Lemma 33. *Let $(K, \text{test}(K), +, \cdot, *, 0, 1, \neg, \lceil \rceil)$ be a KAD. Consider the set of pairs $E = \{(x, t) \in K \times \text{test}(K) \mid t \leq \lceil x \rceil\}$ and $T = \text{test}(K) \times \text{test}(K)$, and define the following operations on elements of E , where $x, y \in K$ and $s, t, u \in \text{test}(K)$.*

$$\begin{aligned}
(x, s) \oplus (y, t) &= (x \sqcup_A y, s \cdot t) \\
(x, s) \odot (y, t) &= (x \sqcap_A y, s \cdot \neg \ulcorner (x \cdot \neg t)) \\
(x, s)^{\otimes} &= (x^{\times_A}, \ulcorner (x^{\times_A} \sqcap_A s)) \\
\overline{(s, s)} &= (\neg s, \neg s) \\
(s, s) \mathbin{\mathbb{M}} (t, t) &= (s \mathbin{\mathbb{F}}_A t, s \mathbin{\mathbb{F}}_A t) \\
\ulcorner (x, s) &= (\ulcorner x, \ulcorner x) \\
(x, s) \mathbin{\mathbb{M}}_{(u, u)} (y, t) &= (x \mathbin{\mathbb{F}}_{Au} y, s \mathbin{\mathbb{F}}_{Au} t)
\end{aligned}$$

Then $(E, T, \oplus, \odot, \otimes, (0, 0), (1, 1), \overline{}, \mathbin{\mathbb{M}}, \ulcorner, \mathbin{\mathbb{M}}_\bullet)$ is a DAD- $\mathbb{F}_\bullet$.

Here is a DAD where the t -decomposition of an element is not necessarily unique. Take the structure constructed in Lemma 33 with relations on the set $\{0, 1\}$ as carrier set K . Take the following relations

$$\begin{aligned}
0 &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & s &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & t &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & 1 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
a &= \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} & b &= \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} & c &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
\end{aligned}$$

and define $\mathbb{T} = (0, 0)$. Then $(c, 0)$ admits nine different (s, s) -decompositions among which we find

$$(c, 0) = \mathbb{T} \mathbin{\mathbb{M}} \mathbb{T} \mathbin{\mathbb{M}} ((a, s) \oplus (b, t)) \quad (39)$$

$$(c, 0) = \mathbb{T} \mathbin{\mathbb{M}} \mathbb{T} \mathbin{\mathbb{M}} ((a, t) \oplus (b, s)) \quad (40)$$

There is a natural interpretation for the construction of Lemma 33. One can view a pair (x, t) as the semantics of a program x having three kinds of initial states. Those that are in t (hence in $\ulcorner x$) always lead to termination and the terminating part of x is $t \cdot x$. Those that are in $\ulcorner x$ but not in t may lead to nontermination or to termination with terminating action $\neg t \cdot x$. Those that are not in $\ulcorner x$ (hence not in t) lead to nontermination. This interpretation is preserved by the operations of the lemma. This means that algebras with elements that have multiple decompositions may have useful applications. This will be the subject of further investigation.

5.2 The First Point of Conjecture 32 Is False

Going back to the example of Sect. 5.1, it is easy to see that the element (a, s) and (b, t) are decomposable, because $(a, s) = (a, s) \mathbin{\mathbb{M}} \mathbb{T} \mathbin{\mathbb{M}} (\mathbb{T} \oplus \mathbb{T})$ and $(b, t) = \mathbb{T} \mathbin{\mathbb{M}} (b, t) \mathbin{\mathbb{M}} (\mathbb{T} \oplus \mathbb{T})$ since $(a, s) = (a, s) \odot (s, s)$ and $(b, t) = (b, t) \odot (s, s)$. Then $(a, s) \oplus (b, t)$ has two possible (s, s) -decompositions, since (see (39) and (40))

$$\begin{aligned}
(a, s) \oplus (b, t) &= (c, 0) = \mathbb{T} \mathbin{\mathbb{M}} \mathbb{T} \mathbin{\mathbb{M}} ((a, s) \oplus (b, t)) \\
(a, s) \oplus (b, t) &= (c, 0) = \mathbb{T} \mathbin{\mathbb{M}} \mathbb{T} \mathbin{\mathbb{M}} ((a, t) \oplus (b, s)) \quad .
\end{aligned}$$

So $(a, s) \oplus (b, t)$ is not decomposable while (a, s) and (b, t) are.

5.3 A Maximal Subalgebra of Decomposable Elements

Proposition 34. *Let $A_{\mathcal{D}}$ be a $DAD\text{-}\mathbb{F}_{\bullet}$. There is a maximal subalgebra (not necessarily unique) of decomposable elements.*

5.4 A True Version of Conjecture 32

Theorem 35. *Let $A_{\mathcal{D}}$ be a $DAD\text{-}\mathbb{F}_{\bullet}$. Let I be a subalgebra of decomposable elements such that $B_{\mathcal{D}} \subseteq I \subseteq A_{\mathcal{D}}$. Then $(I, B_{\mathcal{D}}, +_{\mathcal{D}}, \cdot_{\mathcal{D}}, *_D, \top, 1, \neg, \ulcorner)$ is a KAD.*

Hence we have to consider a subalgebra of decomposable elements to make Conjecture 32 true. Indeed, the first version made mention of *the* subalgebra of decomposable elements while such a subalgebra does not exist in general (see Sect. 5.2). Nevertheless, the fact that there is a maximal subalgebra of decomposable elements (see Sect. 5.3) brings back confidence in the concept of decomposition. In particular, if $A_{\mathcal{D}}$ contains only decomposable elements, then $(A_{\mathcal{D}}, B_{\mathcal{D}}, +_{\mathcal{D}}, \cdot_{\mathcal{D}}, *_D, \top, 1, \neg, \ulcorner)$ is a KAD. It is shown in 3.4 that this construction of a KAD is the inverse of the construction of a KAD-based DAD.

6 Conclusion

It is mentioned in 12 that the *feasible commands* of *command algebras* constitute a DAD. It is equally shown in 5 that the *total* elements of a Demonic refinement algebra 20 constitute a DAD (these two results are intimately related). In both cases, the DADs are KAD-based and thus contain only decomposable elements. An interesting question is therefore whether DADs with nondecomposable elements are relevant for program specification and construction. The remarks made after Lemma 33 above are indications that this is the case. Finally, the question of decidability of $DAD\text{-}\mathbb{F}_{\bullet}$ has not been touched on yet. We have to study 14,16 and see if some ideas can be translated to the universe of demonic algebra.

Acknowledgements

This research was partially supported by NSERC (Natural Sciences and Engineering Research Council of Canada) and FQRNT (Fond québécois de la recherche sur la nature et les technologies).

References

1. Cohen, E.: Separation and reduction. In: Backhouse, R., Oliveira, J.N. (eds.) MPC 2000. LNCS, vol. 1837, pp. 45–59. Springer, Heidelberg (2000)
2. Conway, J.: Regular Algebra and Finite Machines. Chapman and Hall, London (1971)

3. De Carufel, J.L., Desharnais, J.: Demonic algebra with domain. Research report DIUL-RR-0601, Département d'informatique et de génie logiciel, Université Laval, Canada (June 2006),
<http://www.ift.ulaval.ca/~Desharnais/Recherche/RR/DIUL-RR-0601.pdf>
4. De Carufel, J.L., Desharnais, J.: Demonic algebra with domain. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 120–134. Springer, Heidelberg (2006)
5. De Carufel, J.L., Desharnais, J.: On the structure of demonic refinement algebras with enabledness and termination. These proceedings
6. Desharnais, J., Möller, B., Struth, G.: Modal Kleene algebra and applications — A survey —. JORMiCS — Journal on Relational Methods in Computer Science 1, 93–131 (2004)
7. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM Transactions on Computational Logic (TOCL) 7(4), 798–833 (2006)
8. Desharnais, J., Möller, B., Tchier, F.: Kleene under a demonic star. In: Rus, T. (ed.) AMAST 2000. LNCS, vol. 1816, pp. 355–370. Springer, Heidelberg (2000)
9. Desharnais, J., Möller, B., Tchier, F.: Kleene under a modal demonic star. Journal of Logic and Algebraic Programming, Special issue on Relation Algebra and Kleene Algebra 66(2), 127–160 (2006)
10. Hoare, C.A.R., Hayes, I.J., Jifeng, H., Morgan, C.C., Roscoe, A.W., Sanders, J.W., Sorensen, I.H., Spivey, J.M., Sufrin, B.A.: Laws of programming. Communications of the ACM 30(8), 672–686 (1987)
11. Hoare, C.A.R., He, J.: Unifying Theories of Programming. In: International Series in Computer Science, Prentice-Hall, Englewood Cliffs (1998)
12. Höfner, P., Möller, B., Solin, K.: Omega algebra, demonic refinement algebra and commands. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 222–234. Springer, Heidelberg (2006)
13. Hollenberg, M.: Equational axioms of test algebra (1996)
14. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation 110(2), 366–390 (1994)
15. Kozen, D.: Kleene algebra with tests. ACM Transactions on Programming Languages and Systems 19(3), 427–443 (1997)
16. Kozen, D., Smith, F.: Kleene algebra with tests: Completeness and decidability. In: van Dalen, D., Bezem, M. (eds.) CSL 1996. LNCS, vol. 1258, pp. 244–259. Springer, Heidelberg (1997)
17. Mace4. <http://www.cs.unm.edu/~mccune/mace4/>
18. Möller, B.: Lazy Kleene algebra. In: Kozen, D., Shankland, C. (eds.) MPC 2004. LNCS, vol. 3125, pp. 252–273. Springer, Heidelberg (2004)
19. Solin, K., von Wright, J.: Refinement algebra with operators for enabledness and termination. In: Uustalu, T. (ed.) MPC 2006. LNCS, vol. 4014, pp. 397–415. Springer, Heidelberg (2006)
20. von Wright, J.: Towards a refinement algebra. Science of Computer Programming 51, 23–45 (2004)

On the Structure of Demonic Refinement Algebras with Enabledness and Termination

Jean-Lou De Carufel and Jules Desharnais

Département d'informatique et de génie logiciel, Pavillon Adrien-Pouliot,
1065, avenue de la Médecine, Université Laval, Québec, QC, Canada G1V 0A6
jldec1@ift.ulaval.ca, Jules.Desharnais@ift.ulaval.ca

Abstract. The main result of this paper is that every demonic refinement algebra with enabledness and termination is isomorphic to an algebra of ordered pairs of elements of a Kleene algebra with domain and with a divergence operator satisfying a mild condition. Divergence is an operator producing a test interpreted as the set of states from which nontermination may occur.

1 Introduction

Demonic Refinement Algebra (DRA) was introduced by von Wright in [23,24]. It is a variant of Kleene Algebra (KA) and Kleene algebra with tests (KAT) as defined by Kozen [14,15] and of Cohen's omega algebra [3]. DRA is an algebra for reasoning about total correctness of programs and has the positively conjunctive predicate transformers as its intended model. DRA was then extended with enabledness and termination operators by Solin and von Wright [20,21,22], giving an algebra called DRAet in [20] and in this article. The names of these operators reflect their semantic interpretation in the realm of programs and their axiomatisation is inspired by that of the domain operator of Kleene Algebra with Domain (KAD) [8,9]. Further extensions of DRA were investigated with the goal of dealing with both angelic and demonic nondeterminism, one, called **daRAet**, where the algebra has dual join and meet operators and one, called **daRAn**, with a negation operator [19,20]; a generalisation named General Refinement Algebra was also obtained in [24] by weakening the axioms of DRA.

In this paper, we are concerned with the structure of DRAet. The main result is that every DRAet is isomorphic to an algebra of ordered pairs of elements of a KAD with a divergence operator satisfying a mild condition. Divergence is an operator producing a test interpreted as the set of states from which nontermination may occur (see [10] for the divergence operator, and [17,13] for its dual, the convergence operator). It is shown in [13] that a similar algebra of ordered pairs of elements of an omega algebra with divergence is a DRAet; in [17], these algebras of pairs are mapped to weak omega algebras, a related structure. Our result is stronger because (1) it does not require the algebra of pairs to have an ω operator—which is a somewhat surprising result, since DRA has one—(2) it

states not only that the algebras of ordered pairs are DRAs, but that every DRA is isomorphic to such an algebra. A consequence of this result is that every KAD with divergence (satisfying the mild condition) can be embedded in a DRAet.

Section 2 contains the definition of DRAet and properties that can be found in [23,24,20,21,22] or easily derivable from these. We have however decided to invert the partial ordering with respect to the one used by Solin and von Wright. Their order is more convenient when axiomatising predicate transformers, but ours is more in line with the standard KA notation; in particular, this has the effect that the embedded KAD mentioned above keeps its traditional operators after the embedding. Section 3 presents new results about the structure of DRAet, such as the fact that the “bottom part” of the lattice of a DRAet D is a KAD D_K with divergence and the fact that every element x of D can be written as $x = a + t\top$, where $a, t \in D_K$ and t is a test. Section 4 describes the algebra of ordered pairs and proves the results mentioned in the previous paragraph; it also contains an example conveying the intuition behind the formal results. Section 5 discusses prospects for further research. For lack of space, most proofs are omitted; they can be found in [6].

2 Definition of Demonic Refinement Algebra with Enabledness and Termination

We begin with the definition of Demonic Refinement Algebra [23,24].

Definition 1. A demonic refinement algebra (DRA) is a tuple $(D, +, \cdot, *, ^\omega, 0, 1)$ satisfying the following axioms and rules, where $\cdot$ is omitted, as is usually done (i.e., we write xy instead of $x \cdot y$), and where the order $\leq$ is defined by $x \leq y \stackrel{\text{def}}{\iff} x + y = y$. The operators $*$ and $^\omega$ bind equally; they are followed by $\cdot$ and then $+$.

1. $x + (y + z) = (x + y) + z$	9. $(x + y)z = xz + yz$
2. $x + y = y + x$	10. $x^* = xx^* + 1$
3. $x + 0 = x$	11. $xz + y \leq z \Rightarrow x^*y \leq z$
4. $x + x = x$	12. $zx + y \leq z \Rightarrow yx^* \leq z$
5. $x(yz) = (xy)z$	13. $x^\omega = xx^\omega + 1$
6. $1x = x = x1$	14. $z \leq xz + y \Rightarrow z \leq x^\omega y$
7. $0x = 0$	15. $x^\omega = x^* + x^\omega 0$
8. $x(y + z) = xy + xz$	

It is easy to verify that $\leq$ is a partial order and that the axioms state that x^* and x^ω are the least and greatest fixed points, respectively, of $(\lambda z \mid : xz + 1)$. All operators are isotone with respect to $\leq$.

Let

$$\top \stackrel{\text{def}}{=} 1^\omega . \quad (1)$$

One can show

$$x \leq \top , \quad (2)$$

$$\top x = \top , \quad (3)$$

for all $x \in D$. Hence, $\top$ is the top element and a left zero for composition. Other consequences of the axioms are the unfolding (4), sliding (5), denesting (6) and other laws that follow.

$$x^* = x^*x + 1 \quad x^\omega = x^\omega x + 1 \quad (4)$$

$$x(yx)^* = (xy)^*x \quad x(yx)^\omega = (xy)^\omega x \quad (5)$$

$$(x + y)^* = x^*(yx^*)^* \quad (x + y)^\omega = x^\omega(yx^\omega)^\omega \quad (6)$$

$$(x\top)^* = x\top + 1 \quad (x\top)^\omega = x\top + 1 \quad (7)$$

$$(x0)^* = x0 + 1 \quad (x0)^\omega = x0 + 1 \quad (8)$$

An element $t \in D$ that has a complement $\neg t$ satisfying

$$t\neg t = \neg tt = 0 \quad \text{and} \quad t + \neg t = 1 \quad (9)$$

is called a *guard*. Let D_G be the set of guards of D . Then $(D_G, +, \cdot, \neg, 0, 1)$ is a Boolean algebra and it is a maximal one, since every t that has a complement satisfying (9) is in D_G . Properties of guards are similar to those of tests in KAT and KAD.

Every guard t has a corresponding *assertion* t° defined by

$$t^\circ \stackrel{\text{def}}{=} \neg t\top + 1. \quad (10)$$

Guards and assertions are order-isomorphic: $s \leq t \Leftrightarrow t^\circ \leq s^\circ$ for all guards s and t . Thus, assertions form a Boolean algebra too. Assertions have a weaker expressive power than guards and guards cannot be defined in terms of assertions, although the latter are defined in terms of guards.

In the sequel, the symbols p, q, r, s, t , possibly subscripted, denote guards or assertions (which one will be clear from the context). The set of guards and assertions of a DRA D are denoted by D_G and D_A , respectively.

Next, we introduce the enabledness and termination operators [20,21,22]. The definition below is in fact that of [20], because the isolation axiom (Definition 1(15) above) and axioms (14) and (18) below are not included in [21,22].

Definition 2. A demonic refinement algebra with enabledness (DRAe) is a structure $(D, +, \cdot, *, \omega, \ulcorner, 0, 1)$ such that $(D, +, \cdot, *, \omega, 0, 1)$ is a DRA and the enabledness operator $\ulcorner : D \rightarrow D_G$ (mapping elements to guards) satisfies the following axioms, where t is a guard.

$$\ulcorner xx = x \quad (11)$$

$$\ulcorner(tx) \leq t \quad (12)$$

$$\ulcorner(xy) = \ulcorner(x\ulcorner y) \quad (13)$$

$$\ulcorner x\top = x\top \quad (14)$$

A demonic refinement algebra with enabledness and termination (DRAet) is a structure $(D, +, \cdot, *, \omega, \ulcorner, \ulcorner^\omega, 0, 1)$ such that $(D, +, \cdot, *, \omega, \ulcorner, 0, 1)$ is a DRAe and

the termination operator $\ulcorner : D \rightarrow D_A$ (mapping elements to assertions) satisfies the following axioms, where p is an assertion.

$$\ulcorner xx = x \quad (15)$$

$$p \leq \ulcorner(px) \quad (16)$$

$$\ulcorner(xy) = \ulcorner(x\ulcorner y) \quad (17)$$

$$\ulcorner x0 = x0 \quad (18)$$

The termination operator is defined by four axioms in Definition 2 in order to exhibit its similarity with the enabledness operator. It turns out however that Axioms (15), (16) and (17) can be dropped, because they follow from Axiom (18). It is also shown in [20] that $\ulcorner x0 = x0 \Leftrightarrow \ulcorner x = x0 + 1$. Thus (15) to (18) are equivalent to $\ulcorner x = x0 + 1$ and it looks like the termination operator might be defined by $\ulcorner x \stackrel{\text{def}}{=} x0 + 1$, a possibility that is also mentioned in [21,22]. However, Solin and von Wright remark that this is not possible unless it is known that $x0 + 1$ is an assertion; it is shown in [19,20] that $x0 + 1$ is an assertion in **daRAet**. We show in Sect. 3 that this is the case in **DRAe** too.

The following are laws of enabledness.

$$\ulcorner t = t \quad (19)$$

$$\ulcorner \top = 1 \quad (20)$$

$$\ulcorner(x + y) = \ulcorner x + \ulcorner y \quad (21)$$

$$\ulcorner(tx) = t\ulcorner x \quad (22)$$

$$\neg \ulcorner xx = 0 \quad (23)$$

$$\ulcorner x = 0 \Leftrightarrow x = 0 \quad (24)$$

$$\neg \ulcorner(xt)x = \neg \ulcorner(xt)x \neg t \quad (25)$$

In addition, both enabledness and termination are isotone. The first three axioms of enabledness, (11), (12) and (13), are exactly the axioms of the domain operator in KAD. We do not explain at this stage the intuitive meaning of enabledness and termination. This will become clear in Sect. 4 after the introduction of the representation of DRA by algebras of pairs.

In DRA, there seems to be no way to recover by an explicit definition the guard corresponding to a given assertion. This becomes possible in **daRA** and **daRAn** [19,20]. We show in Sect. 3 that it is also possible in **DRAe**.

3 Structure of Demonic Refinement Algebras with Enabledness and Termination

This section contains new results about **DRAe** and **DRAet**. It is first shown that in **DRAe**, guards can be defined in terms of assertions and that the termination operator can be explicitly defined in **DRAe** rather than being implicitly defined by Axioms (15) to (18). This means that every **DRAe** is also a **DRAet**, so

that the two concepts are equivalent. After introducing KAD and the divergence operator, we show that every DRAet D contains an embedded KAD D_K with divergence and that every element of D can be decomposed into its terminating and nonterminating parts, both essentially expressed by means of D_K .

Proposition 3. *Let D be a DRAe and $\diamond : D_A \rightarrow D_G$ be the function defined by*

$$p^\diamond \stackrel{\text{def}}{=} \neg \lceil p0 \rceil . \quad (26)$$

Then, for any assertion p and guard t

1. $p^\diamond$ is a guard with complement $\lceil p0 \rceil$,
2. $t^{\circ\diamond} = t$,
3. $p^{\circ\diamond} = p$. Combined with the previous item, this says that $\circ$ and $\diamond$ are dual isomorphisms.

Now let the operators $\neg : D_A \rightarrow D_A$ and $\sqcap : D_A \times D_A \rightarrow D_A$ be defined by

$$\neg p \stackrel{\text{def}}{=} (\neg(p^\diamond))^\circ \quad \text{and} \quad (27)$$

$$p \sqcap q \stackrel{\text{def}}{=} \neg(\neg p + \neg q) , \quad (28)$$

for any assertions p and q .

Proposition 4. *For a given DRAe, the structures*

$$(D_A, \sqcap, +, \neg, \top, 1) \quad \text{and} \quad (D_G, +, \cdot, \neg, 0, 1)$$

are isomorphic Boolean algebras, with the isomorphism given either by $\circ$ or $\diamond$.

This is of course consistent with the remark about the order-isomorphism of assertions and guards made in the previous section. Since inverting the order of a Boolean algebra yields another Boolean algebra, $(D_A, +, \sqcap, \neg, 1, \top)$ is also a Boolean algebra and it is ordered by the DRAe ordering $\leq$.

Lemma 5. *In a DRAe, $x0 + 1$ is an assertion.*

Proof. Using in turn Definition [1\(7\)](#), [\(14\)](#), double negation (applicable since $\lceil x0 \rceil$ is a guard) and [\(10\)](#), we get

$$x0 + 1 = x0\top + 1 = \lceil x0 \rceil\top + 1 = \neg\neg\lceil x0 \rceil\top + 1 = (\neg\lceil x0 \rceil)^\circ .$$

Thus, $x0 + 1$ is an assertion and, by Proposition [3](#), it uniquely corresponds to the guard $\neg\lceil x0 \rceil$. $\square$

This means that it is now possible to give an explicit definition of $\top$.

Definition 6. *For a given DRAe D , the termination operator $\top : D \rightarrow D_A$ is defined by $\top x \stackrel{\text{def}}{=} x0 + 1$.*

By the results of Solin and von Wright mentioned in Sect. 2, the termination operator satisfies Axioms (I5) to (I8).

We now recall the definition of KAD [8,9].

Definition 7. A Kleene Algebra with Domain (KAD) is a structure $(K, +, \cdot, *, \top, 0, 1)$ satisfying all axioms of DRAe, except those involving ω (i.e., Definition 7 (I3, I4, I5)) and $\top$ (i.e., (I4)), with the additional axiom that 0 is a right zero of composition:

$$x0 = 0 . \quad (29)$$

The range of the domain operator $\top$ is a Boolean subset of K denoted by $\text{test}(K)$ whose elements are called tests. Tests satisfy the laws of guards in a DRAe [9].

The standard signature of KAT and KAD includes a sort $B \subseteq K$ of tests and a negation operator on B [15,8,9]. We have chosen not to include them here in order to have a signature close to that of DRAe. In KAT, B can be any Boolean subset of K , but in KAD, the domain operator forces B to be the maximal Boolean subset of elements below 1 [9]. Thus, the definition of tests in KAD given above imposes the same constraints as that of guards in DRA given in Sect. 2.

The domain operator satisfies the following inductive law (as does the enabledness operator of DRAe) [9]:

$$\top(xt) + s \leq t \Rightarrow \top(x^*s) \leq t . \quad (30)$$

In a given KAD, the greatest fixed point $(\nu t \mid t \in \text{test}(K) : \top(xt))$, may or may not exist. This fixed point plays an important role in the sequel. We will denote it by ∇x and axiomatise it by

$$\nabla x \leq \top(x\nabla x) , \quad (31)$$

$$t \leq \top(xt) \Rightarrow t \leq \nabla x . \quad (32)$$

∇x is called the *divergence of x* [10] and this test is interpreted as the set of states from which nontermination is possible. The negation of ∇x corresponds to what is known as the *halting predicate* in the modal μ -calculus [12]. The operator ∇ binds stronger than any binary operator but weaker than any unary operator. Among the properties of divergence, we note

$$\nabla x = \top(x\nabla x) , \quad (33)$$

$$x\nabla x = \nabla xx\nabla x , \quad (34)$$

$$\neg\nabla xx = \neg\nabla xx\neg\nabla x , \quad (35)$$

$$\nabla(tx) \leq t , \quad (36)$$

$$x \leq y \Rightarrow \nabla x \leq \nabla y . \quad (37)$$

Proposition 8. In a KAD K where ∇x exists for every $x \in K$, $\top(x^*s) + \nabla x$ is a fixed point of $f(t) \stackrel{\text{def}}{=} \top(xt) + s$ and

$$t \leq \top(xt) + s \Rightarrow t \leq \top(x^*s) + \nabla x , \quad (38)$$

that is, $\top(x^*s) + \nabla x$ is the greatest fixed point of f .

The proof of this proposition is given in [10].

In the sequel, we denote by D_K the following set of elements of a DRAe D :

$$D_K \stackrel{\text{def}}{=} \{x \in D \mid x0 = 0\} . \quad (39)$$

Theorem 9. *Let D be a DRAe. Then $(D_K, +, \cdot, *, \top, 0, 1)$ is a KAD in which ∇x exists for all x . In addition, the set of tests of D_K is the set of guards D_G and*

$$\nabla x = \top(x^\omega 0) , \quad (40)$$

$$\nabla x = 0 \wedge z \leq xz + y \Rightarrow z \leq x^*y . \quad (41)$$

Proof. The elements of D_K satisfy all axioms of KAD, including (29). All we need to prove in order to show that D_K is a KAD is that it is closed under the operations of KAD. First, D_K contains 1 and 0, since $10 = 0$ and $00 = 0$. Next, if t is a guard, then $t \in D_K$, since $t0 \leq 10 = 0$. Thus, guards are the tests of D_K and form a Boolean algebra with the operations $+$, $\cdot$ and $\neg$. This implies $\top x \in D_K$ for all x , since $\top x$ is a guard. Finally, for the remaining operations, we have the following, where $x0 = 0$ and $y0 = 0$ are assumed, due to (39):

- $(x + y)0 = x0 + y0 = 0$ by Definition 1(9,4);
- $xy0 = x0 = 0$;
- $x^*0 \leq 0 \Leftarrow x0 + 0 \leq 0 \Leftarrow \text{true}$ by Definition 1(11,4).

For the proof of (40) and (41), see [6]. □

Theorem 10. *Let D be a DRAe and t be a guard in D (hence in D_K). Then*

$$\top(x0)x = \top(x0)\top = x0 , \quad (42)$$

$$x = \neg\top(x0)x + \top(x0)\top , \quad (43)$$

$$x = \neg\top(x0)x + x0 . \quad (44)$$

Every $x \in D$ can be written as $x = a + t\top$, where $a, t \in D_K$ and $ta = 0$.

Proof. We start with (42). The refinement $\top(x0)x \leq \top(x0)\top$ follows from $x \leq \top$. The other refinement and the equality follow from (14), Definition 1(7), (11) and $0 \leq 1$: $\top(x0)\top = x0\top = x0 = \top(x0)x0 \leq \top(x0)x$. This is used in the proof of (43), together with the Boolean algebra of guards and Definition 1(9): $x = (\neg\top(x0) + \top(x0))x = \neg\top(x0)x + \top(x0)x = \neg\top(x0)x + \top(x0)\top$. Equation (44) follows from (43), (14) and Definition 1(7). And $\neg\top(x0)x \in D_K$, since $\neg\top(x0)x0 = 0$ by (23), so that, by (43), $x = a + t\top$, with $a \stackrel{\text{def}}{=} \neg\top(x0)x \in D_K$ and $t \stackrel{\text{def}}{=} \top(x0) \in D_K$ satisfying $ta = 0$ by Boolean algebra and Definition 1(7). □

In (44), $x0$ is the *infinite* or *nonterminating* part of x and $\neg\top(x0)x$ is its *finite* or *terminating* part [16]. The possibility to write any element of D as $a + t\top$ with $a, t \in D_K$ and $ta = 0$ means that both the terminating part a and the nonterminating part $t\top$ are essentially described by the elements a and t of the KAD D_K . Under this form, we already foresee the algebra of ordered pairs (a, t) of Sect. 4.

Another part of the DRAe structure worth mentioning is the set

$$D_D \stackrel{\text{def}}{=} \{x \in D \mid x\top = \top\} . \quad (45)$$

This set contains all the assertions, since for any guard t , $t^\circ \top = (\neg t \top + 1) \top = \top$ (see (10)). Its elements are the *total* or *nonmiraculous* elements and they satisfy $\lceil x = 1$. As already remarked in [13], the substructure D_D of D is a *Demonic Algebra with Domain* (DAD) in the sense of [45,7]. The set D_D is the image of D_K by the transformation $\phi(x) \stackrel{\text{def}}{=} x + \neg \lceil x \top$. The ordering $\sqsubseteq$ of DAD satisfies $x \sqsubseteq y \Leftrightarrow \phi(x) \leq \phi(y)$. Now let $\psi(x) = \neg \lceil x 0 x$, where $x \in D_D$. It is easy to prove that ψ is the inverse of ϕ . The following properties can then be derived. In these, $x, y \in D_K$. The notation for the demonic operators is that of [45,7]. The demonic operators of DAD are concerned only with the terminating part of the elements of D_D . For each operator, the first $\stackrel{\text{def}}{=}$ transformation is obtained by calculating the image in D_D of x and y , using ϕ . An operation of D is then applied and, finally, the terminating part of the result is kept, using ψ . The final expression given for each operator is exactly the expression defining KAD-based demonic operators in [45,7].

1. Demonic join: $x \sqcup y \stackrel{\text{def}}{=} \psi(\phi(x) + \phi(y)) = \lceil x \rceil y(x + y)$.
2. Demonic composition: $x \circ y \stackrel{\text{def}}{=} \psi(\phi(x)\phi(y)) = \neg \lceil x \neg \lceil y xy$.
3. Demonic star: $x^\times \stackrel{\text{def}}{=} \psi((\phi(x))^*) = x^* \sqcap x$.
4. Demonic negation: $\neg t \stackrel{\text{def}}{=} \psi(\neg(\phi(t))) = \neg t$.
5. Demonic domain: $\lceil x \stackrel{\text{def}}{=} \psi(\lceil(\phi(x))) = \lceil x$.

However, unlike what is shown for KAD in Theorem 13 below, not every DAD can be embedded in a DRA, because not every DAD is the image of a KAD.

4 A Demonic Refinement Algebra of Pairs

This section contains the main theorem of the article (Theorem 13), about the isomorphism between any DRAe and an algebra of ordered pairs. We first define this algebra of pairs, show that it is a DRAe and then prove Theorem 13. At the end of the section, Example 14 provides a semantically intuitive understanding of the results of the paper.

Definition 11. *Let K be a KAD such that*

$$\forall x \text{ exists for all } x \in K \quad \text{and} \quad \forall x = 0 \wedge z \leq xz + y \Rightarrow z \leq x^*y. \quad (46)$$

Define the set of ordered pairs P by

$$P \stackrel{\text{def}}{=} \{(x, t) \mid x \in K \wedge t \in \text{test}(K) \wedge tx = 0\}.$$

We define the following operations on P .

1. $(x, s) \oplus (y, t) \stackrel{\text{def}}{=} (\neg(s + t)(x + y), s + t)$
2. $(x, s) \odot (y, t) \stackrel{\text{def}}{=} (\neg \lceil xt xy, s + \lceil xt)$
3. $(x, t)^\circ \stackrel{\text{def}}{=} (\neg \lceil x^* t x^*, \lceil x^* t)$

4. $(x, t)^{\tilde{\omega}} \stackrel{\text{def}}{=} (\neg\lceil x^*t \rceil \neg \nabla x x^*, \lceil x^*t \rceil + \nabla x)$
5. $\lceil x, t \rceil \stackrel{\text{def}}{=} (\lceil x + t, 0 \rceil)$

It is easy to verify that the result of each operation is a pair of P . The condition on pairs can be expressed in many equivalent ways

$$tx = 0 \Leftrightarrow t \leq \neg\lceil x \rceil \Leftrightarrow \lceil x \rceil \leq \neg t \Leftrightarrow \neg tx = x \Leftrightarrow \neg t\lceil x \rceil = \lceil x \rceil, \quad (47)$$

by (24) for KAD, (22) for KAD, (11) for KAD and Boolean algebra. The programming interpretation of a pair (x, t) is that t denotes the set of states from which nontermination is possible, while x denotes the terminating computations.

If K were a complete lattice (in particular, if K were finite), only the existence of ∇x would be needed to get all of (46) [1]. We do not know if this is the case for an arbitrary KAD. Note that D_K satisfies (46), by Theorem 9.

Theorem 12. *The algebra $(P, \oplus, \odot, \otimes, \tilde{\omega}, \lceil, \rceil, (0, 0), (1, 0))$ is a DRAe. Moreover,*

1. $(x, s) \sqsubseteq (y, t) \Leftrightarrow s \leq t \wedge \neg tx \leq y$, where $(x, s) \sqsubseteq (y, t) \stackrel{\text{def}}{\Leftrightarrow} (x, s) \oplus (y, t) = (y, t)$,
2. the top element is $(0, 1)$,
3. guards have the form $(t, 0)$, and $\neg(t, 0) = (\neg t, 0)$,
4. the assertion corresponding to the guard $(t, 0)$ is $(t, \neg t)$,
5. $\neg(t, \neg t) = (\neg t, t)$,
6. $\lceil x, t \rceil = (\neg t, t)$.

And now the main theorem.

Theorem 13. 1. Every DRAe is isomorphic to an algebra of ordered pairs as in Definition 11. The isomorphism is given by $\phi(x) \stackrel{\text{def}}{=} (\neg\lceil x0 \rceil x, \lceil x0 \rceil)$, with inverse $\psi((x, t)) \stackrel{\text{def}}{=} x + t\top$.
 2. Every KAD K satisfying (46) can be embedded in a DRAe D in such a way that D_K is the image of K by the embedding.

Proof. 1. Let D be a DRAe. The sub-Kleene algebra $(D_K, +, \cdot, *, \lceil, \rceil, 0, 1)$ of D satisfies (46), by Theorem 9. Use D_K to construct an algebra of pairs $(P, \oplus, \odot, \otimes, \tilde{\omega}, \lceil, \rceil, (0, 0), (1, 0))$ as per Definition 11. We first show that ψ is the inverse of ϕ , so that they both are bijective functions.

$$\begin{aligned}
 \text{(a)} \quad & \psi(\phi(x)) \\
 &= \psi((\neg\lceil x0 \rceil x, \lceil x0 \rceil)) \\
 &= \neg\lceil x0 \rceil x + \lceil x0 \rceil \top \\
 &= \langle \text{(14)} \ \& \ \text{Definition 11(7)} \rangle \\
 & \quad \neg\lceil x0 \rceil x + x0 \\
 &= \langle \text{(44)} \rangle \\
 & \quad x
 \end{aligned}$$

$$\begin{aligned}
\text{(b)} \quad & \phi(\psi((x, t))) \\
&= \phi(x + t\top) \\
&= (\neg\top((x + t\top)0)(x + t\top), \top((x + t\top)0)) \\
&= \langle \text{Definition } \textcolor{red}{11(9)} \text{ \& } \textcolor{red}{(3)} \rangle \\
&\quad (\neg\top(x0 + t\top)(x + t\top), \top(x0 + t\top)) \\
&= \langle \text{Since } x \in D_K, x0 = 0 \text{ by } \textcolor{red}{(39)} \text{ \& } \text{Definition } \textcolor{red}{11(3)} \rangle \\
&\quad (\neg\top(t\top)(x + t\top), \top(t\top)) \\
&= \langle \textcolor{red}{(13)} \text{ \& } \textcolor{red}{(20)} \text{ \& } \text{Definition } \textcolor{red}{11(6)} \text{ \& } \textcolor{red}{(19)} \rangle \\
&\quad (\neg t(x + t\top), t) \\
&= \langle \text{Definition } \textcolor{red}{11(8,7,3)} \text{ \& } \text{Boolean algebra \& } \neg tx = x \text{ by } \textcolor{red}{(47)} \rangle \\
&\quad (x, t)
\end{aligned}$$

2. What remains to show is that ϕ preserves the operations. Since ψ is the inverse of ϕ , it is equivalent to show that ψ preserves the operations and this is what we do (it is somewhat simpler).

$$\begin{aligned}
\text{(a)} \quad & \psi((x, s) \oplus (y, t)) \\
&= \psi((\neg(s + t)(x + y), s + t)) \\
&= \neg(s + t)(x + y) + (s + t)\top \\
&= \langle \text{Boolean algebra \& } \text{Definition } \textcolor{red}{11(8,9)} \rangle \\
&\quad \neg t\neg sx + \neg s\neg ty + s\top + t\top \\
&= \langle sx = 0 \text{ \& } ty = 0 \text{ \& } \textcolor{red}{(47)} \text{ \& } tx \leq t\top \text{ \& } sy \leq s\top \rangle \\
&\quad \neg tx + tx + \neg sy + sy + s\top + t\top \\
&= \langle \text{Definition } \textcolor{red}{11(9,2,6)} \text{ \& } \text{Boolean algebra} \rangle \\
&\quad x + s\top + y + t\top \\
&= \psi((x, s)) + \psi((y, t)) \\
\text{(b)} \quad & \psi((x, s) \odot (y, t)) \\
&= \psi((\neg\top(xt)xy, s + \top(xt))) \\
&= \neg\top(xt)xy + (s + \top(xt))\top \\
&= \langle \text{Definition } \textcolor{red}{11(9)} \text{ \& } \top(xt)xy \leq \top(xt)\top \rangle \\
&\quad \neg\top(xt)xy + \top(xt)xy + s\top + \top(xt)\top \\
&= \langle \text{Definition } \textcolor{red}{11(9,6)} \text{ \& } \text{Boolean algebra \& } \textcolor{red}{(14)} \rangle \\
&\quad xy + s\top + xt\top \\
&= \langle \text{Definition } \textcolor{red}{11(9,8)} \text{ \& } \textcolor{red}{(3)} \rangle \\
&\quad (x + s\top)(y + t\top) \\
&= \psi((x, s)) \cdot \psi((y, t))
\end{aligned}$$

$$\begin{aligned}
(c) \quad & \psi((x, t)^{\otimes}) \\
&= \psi((\neg(\ulcorner x^*t \urcorner)x^*, \ulcorner x^*t \urcorner)) \\
&= \neg(\ulcorner x^*t \urcorner)x^* + \ulcorner x^*t \urcorner\top \\
&= \langle \ulcorner x^*t \urcorner x^* \leq \ulcorner x^*t \urcorner\top \rangle \\
&\quad \neg(\ulcorner x^*t \urcorner)x^* + \ulcorner x^*t \urcorner x^* + \ulcorner x^*t \urcorner\top \\
&= \langle \text{Definition } \textcolor{red}{11(9.6)} \text{ \& Boolean algebra \& } \textcolor{red}{(14)} \rangle \\
&\quad x^* + x^*t\top \\
&= \langle \text{Definition } \textcolor{red}{11(8.2.6)} \text{ \& } \textcolor{red}{(7)} \rangle \\
&\quad x^*(t\top)^* \\
&= \langle \textcolor{red}{(3)} \rangle \\
&\quad x^*(t\top x^*)^* \\
&= \langle \textcolor{red}{(6)} \rangle \\
&\quad (x + t\top)^* \\
&= (\psi((x, t)))^* \\
\\
(d) \quad & \psi((x, t)^{\tilde{\omega}}) \\
&= \psi((\neg(\ulcorner x^*t \urcorner)\neg\forall x x^*, \ulcorner x^*t \urcorner + \forall x)) \\
&= \neg(\ulcorner x^*t \urcorner)\neg\forall x x^* + (\ulcorner x^*t \urcorner + \forall x)\top \\
&= \langle \text{De Morgan \& } (\ulcorner x^*t \urcorner + \forall x)x^* \leq (\ulcorner x^*t \urcorner + \forall x)\top \rangle \\
&\quad \neg(\ulcorner x^*t \urcorner + \forall x)x^* + (\ulcorner x^*t \urcorner + \forall x)x^* + (\ulcorner x^*t \urcorner + \forall x)\top \\
&= \langle \text{Definition } \textcolor{red}{11(9.6)} \text{ \& Boolean algebra \& } \textcolor{red}{(40)} \rangle \\
&\quad x^* + \ulcorner x^*t \urcorner\top + \ulcorner x^\omega 0 \urcorner\top \\
&= \langle \textcolor{red}{(14)} \text{ \& Definition } \textcolor{red}{11(7)} \text{ \& } x^\omega 0 = x^\omega t 0\top \rangle \\
&\quad x^* + x^*t\top + x^\omega 0 + x^\omega 0t\top \\
&= \langle \text{Definition } \textcolor{red}{11(2.9.15)} \rangle \\
&\quad x^\omega + x^\omega t\top \\
&= \langle \text{Definition } \textcolor{red}{11(6.8.2)} \text{ \& } \textcolor{red}{(7)} \rangle \\
&\quad x^\omega(t\top)^\omega \\
&= \langle \textcolor{red}{(6)} \text{ \& } \textcolor{red}{(3)} \rangle \\
&\quad (x + t\top)^\omega \\
&= (\psi((x, t)))^\omega \\
\\
(e) \quad & \psi(\ulcorner x, t \urcorner) \\
&= \psi(\ulcorner x + t, 0 \urcorner) \\
&= \ulcorner x + t + 0 \urcorner\top
\end{aligned}$$

$$\begin{aligned}
&= \langle \text{Definition } \textcolor{red}{11(73)} \rangle \\
&\quad \ulcorner x + t \\
&= \langle \textcolor{red}{(21)} \ \& \ \textcolor{red}{(13)} \ \& \ \textcolor{red}{(20)} \ \& \ \text{Definition } \textcolor{red}{11(6)} \rangle \\
&\quad \urcorner (x + t \top) \\
&= \urcorner (\psi((x, t))) \\
\text{(f)} &\text{ By definition of } \psi \text{ and Definition } \textcolor{red}{11(73)}, \psi((0, 0)) = 0 + 0\top = 0. \\
\text{(g)} &\text{ By definition of } \psi \text{ and Definition } \textcolor{red}{11(73)}, \psi((1, 0)) = 1 + 0\top = 1. \quad \square
\end{aligned}$$

Example 14. Figure [1](#) may help visualising some of the results of the paper. It displays the DRAe of ordered pairs built from the algebra of all 16 relations over the set $\{\bullet, \circ\}$. The following abbreviations are used: $\mathbf{a} = \{(\bullet, \circ)\}$, $\mathbf{b} = \{(\circ, \bullet)\}$, $\mathbf{s} = \{(\bullet, \bullet)\}$, $\mathbf{t} = \{(\circ, \circ)\}$, $\mathbf{0} = \{\}$, $\mathbf{\top} = \mathbf{a} + \mathbf{b} + \mathbf{s} + \mathbf{t}$, $\mathbf{1} = \mathbf{s} + \mathbf{t}$, $\mathbf{\bar{1}} = \mathbf{a} + \mathbf{b}$. The guards are $(0, 0)$, $(\mathbf{s}, 0)$, $(\mathbf{t}, 0)$, $(1, 0)$ and the assertions are $(1, 0)$, $(\mathbf{t}, \mathbf{s})$, $(\mathbf{s}, \mathbf{t})$, $(0, 1)$. The conjunctive predicate transformer f corresponding to a pair (x, t) is given by $f(s) \stackrel{\text{def}}{=} \neg t \neg \urcorner (x \neg s)$. In words, a transition by x is guaranteed to reach a state in s if the initial state cannot lead to nontermination ($\neg t$) and it is not possible for x to reach a state that is not in s ($\neg \urcorner (x \neg s)$).

Going back to Figure [1](#), we see that the terminating elements, that is, those of the form $(x, 0)$, form a Kleene algebra, in this case a relation algebra isomorphic to the full algebra of relations over $\{\bullet, \circ\}$. For these terminating elements, $\urcorner (x, 0) = (\urcorner x, 0)$ (by Definition [11](#)), so that enabledness on pairs directly corresponds to the domain operator on the first component relation.

Another subset of the pairs is identified as the *nonmiraculous elements*, or *demonic algebra*, in the figure. This subset forms a demonic algebra [\[45, 7\]](#). Its pairs are total, that is, $\urcorner (x, t) = (\urcorner x + t, 0) = (1, 0)$ (the identity element on pairs). From any starting state, (x, t) is *enabled*, in the sense that it either leads to a result or to nontermination. The termination operator applied to (x, t) gives $\urcorner (x, t) = (\neg t, t)$ (Theorem [12\(6\)](#)). This is interpreted as saying that termination is guaranteed for initial states in $\neg t$. In the demonic algebra of [\[45, 7\]](#), the demonic domain of x , $\urcorner x$, is equal to $\neg t$, so that the termination operator and demonic domain correspond on the subset of nonmiraculous elements.

Some elements are nonterminating, some are miraculous, and some are both, such as $(0, \mathbf{t})$. This element does not terminate for initial states in $\mathbf{t}$ (here, $\{\circ\}$) and terminates for states in $\neg \mathbf{t}$ while producing no result (due to the first component being 0).

Instead of viewing pairs as the representation of programs, we can view them as specifications. The weakest specification is $(0, 1)$ at the top of the lattice. It does not even require termination for a single initial state. Lower down, there is the *havoc* element $(\mathbf{\top}, 0)$. As a specification, it requires termination, but arbitrary final states are assigned to initial states. Still lower, there is the identity element $(1, 0)$. It requires termination and assigns a single final state to each initial state. The least element of the lattice, $(0, 0)$ also requires termination, but it is a specification so strong that it assigns no final state to any initial state; we could say it is a contradictory specification.

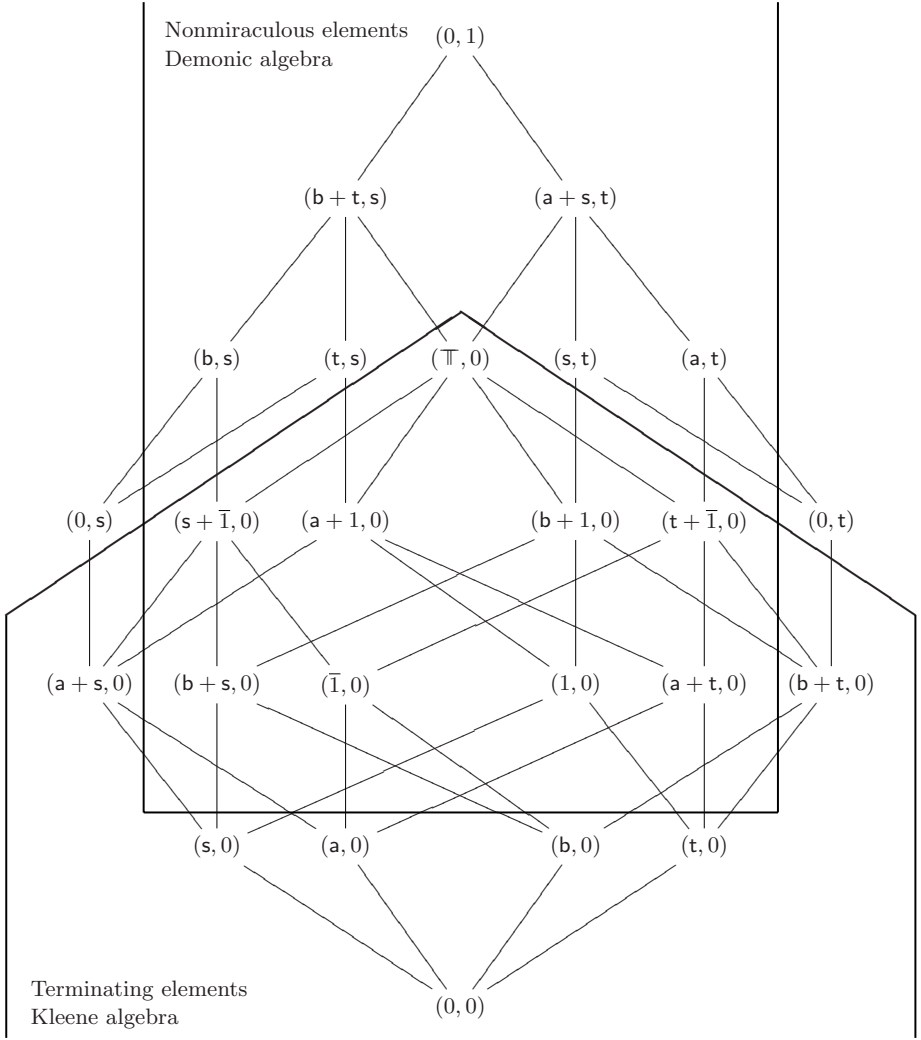


Fig. 1. A demonic refinement algebra of ordered pairs

5 Conclusion

The main theorem of the article, Theorem 13, provides an alternative, equivalent way to view a DRAe as an algebra of ordered pairs. This view, or the related decomposition of any element x of a DRAe as $x = a + t\top$ (Theorem 10), offers an intuitive grasp of the underlying programming concepts that is easier to understand than the predicate transformer model of DRAe for the relationally minded (this may explain why pair-based representations have been used numerous times, such as in [2, 11, 13, 17, 18], to cite just a few).

It is asserted in [10] that the divergence operator often provides a more convenient description of nontermination than the ω operator of omega algebra. Theorem 13 brings some weight to this assertion, because DRAe, although it has an ω operator (different from that of omega algebra, though), is equivalent to an algebra of ordered pairs of elements of a KAD with divergence and without an ω operator.

A side effect of Theorem 13 is that the complexity of the theory of DRAe is at most that of KAD with a divergence operator satisfying the implication in 46 (this complexity is unknown at the moment).

As future work, we plan to look at the variants of DRAe mentioned in the introduction to see if similar results can be obtained.

Acknowledgements

We thank Georg Struth and the anonymous referees for their helpful comments. This research was partially supported by NSERC (Natural Sciences and Engineering Research Council of Canada) and FQRNT (Fond québécois de la recherche sur la nature et les technologies).

References

1. Backhouse, R.: Galois connections and fixed point calculus. In: Backhouse, R., Crole, R.L., Gibbons, J. (eds.) *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*. LNCS, vol. 2297, pp. 89–150. Springer, Heidelberg (2002)
2. Berghammer, R., Zierer, H.: Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science* 43(2–3), 123–147 (1986)
3. Cohen, E.: Separation and reduction. In: Backhouse, R., Oliveira, J.N. (eds.) *MPC 2000*. LNCS, vol. 1837, pp. 45–59. Springer, Heidelberg (2000)
4. De Carufel, J.L., Desharnais, J.: Demonic algebra with domain. Research report DIUL-RR-0601, Département d’informatique et de génie logiciel, Université Laval, Canada (June 2006), <http://www.ift.ulaval.ca/~Desharnais/Recherche/RR/DIUL-RR-0601.pdf>
5. De Carufel, J.L., Desharnais, J.: Demonic algebra with domain. In: Schmidt, R.A. (ed.) *RelMiCS/AKA 2006*. LNCS, vol. 4136, pp. 120–134. Springer, Heidelberg (2006)
6. De Carufel, J.L., Desharnais, J.: On the structure of demonic refinement algebras. Research report DIUL RR-0802, Département d’informatique et de génie logiciel, Université Laval, Québec, Canada (January 2008), <http://www.ift.ulaval.ca/~Desharnais/Recherche/RR/DIUL-RR-0802.pdf>
7. De Carufel, J.L., Desharnais, J.: Latest news about demonic algebra with domain. These proceedings
8. Desharnais, J., Möller, B., Struth, G.: Modal Kleene algebra and applications — A survey—. *JoRMiCS — Journal on Relational Methods in Computer Science* 1, 93–131 (2004)
9. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. *ACM Transactions on Computational Logic (TOCL)* 7(4), 798–833 (2006)

10. Desharnais, J., Möller, B., Struth, G.: Algebraic notions of termination. Research report 2006-23, Institut für Informatik, Universität Augsburg, Germany (October 2006)
11. Doornbos, H.: A relational model of programs without the restriction to Egli-Milner-monotone constructs. In: PROCOMET 1994: Proceedings of the IFIP TC2/WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi, pp. 363–382. North-Holland, Amsterdam (1994)
12. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
13. Höfner, P., Möller, B., Solin, K.: Omega algebra, demonic refinement algebra and commands. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 222–234. Springer, Heidelberg (2006)
14. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation* 110(2), 366–390 (1994)
15. Kozen, D.: Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems* 19(3), 427–443 (1997)
16. Möller, B.: Kleene getting lazy. *Science of Computer Programming* 65, 195–214 (2007)
17. Möller, B., Struth, G.: **wp** is **wlp**. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) RelMiCS 2005. LNCS, vol. 3929, pp. 200–211. Springer, Heidelberg (2006)
18. Parnas, D.L.: A generalized control structure and its formal definition. *Communications of the ACM* 26(8), 572–581 (1983)
19. Solin, K.: On two dually nondeterministic refinement algebras. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 373–387. Springer, Heidelberg (2006)
20. Solin, K.: Abstract Algebra of Program Refinement. PhD thesis, Turku Center for Computer Science, University of Turku, Finland (2007)
21. Solin, K., von Wright, J.: Refinement algebra extended with operators for enabledness and termination. Technical Report 658, Turku Center for Computer Science, University of Turku, Finland, TUCS Technical Report (January 2005)
22. Solin, K., von Wright, J.: Refinement algebra with operators for enabledness and termination. In: Uustalu, T. (ed.) MPC 2006. LNCS, vol. 4014, pp. 397–415. Springer, Heidelberg (2006)
23. von Wright, J.: From Kleene algebra to refinement algebra. Technical Report 450, Turku Center for Computer Science (March 2002)
24. von Wright, J.: Towards a refinement algebra. *Science of Computer Programming* 51, 23–45 (2004)

Multi-objective Problems in Terms of Relational Algebra

Florian Diedrich^{1,*}, Britta Kehden¹, and Frank Neumann²

¹ Institut für Informatik, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, 24098 Kiel, Germany
`{fdi,bk}@informatik.uni-kiel.de`

² Algorithms and Complexity,
Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
`fne@mpi-inf.mpg.de`

Abstract. Relational algebra has been shown to be a powerful tool for solving a wide range of combinatorial optimization problems with small computational and programming effort. The problems considered in recent years are single- objective ones where one single objective function has to be optimized. With this paper we start considerations on the use of relational algebra for multi-objective problems. In contrast to single-objective optimization multiple objective functions have to be optimized at the same time usually resulting in a set of different trade-offs with respect to the different functions. On the one hand, we examine how to solve the mentioned problem exactly by using relational algebraic programs. On the other hand, we address the problem of objective reduction that has recently been shown to be NP-hard. We propose an exact algorithm for this problem based on relational algebra. Our experimental results show that this algorithm drastically outperforms the currently best one.

1 Introduction

Many real-world problems involve optimization of several objective functions simultaneously. For such multi-objective optimization problems usually there is not a single optimal function value for which a corresponding solution should be computed but a set of different trade- offs with respect to the different functions. This set of objective vectors is called the Pareto front of the given problem. Even for two objective functions the Pareto front may be exponential in the problem dimension. This is one reason for the assumption that multi-objective problems are in most cases harder to solve than single-objective ones. Other results from complexity theory support this claim as simple single- objective combinatorial optimization problems such as minimum spanning trees or shortest path become

* Research supported in part by a grant “DAAD Doktorandenstipendium” of the German Academic Exchange Service and in part by EU research project AEOLUS, “Algorithmic Principles for Building Efficient Overlay Computers”, EU contract number 015964.

NP-hard when two functions on the edges should be optimized at the same time [9]. Often optimizing just one of the given objective functions is an NP-hard task. Such problems occur frequently in network design problems where e. g. one task is to minimize the maximum degree of a spanning tree [10,13,16,18]. Another well-known example is the multi-objective knapsack problem [20] where the task is to solve different knapsack problems simultaneously. This problem is a generalization of the classical knapsack problem which belongs to the oldest problems in combinatorial optimization; see the textbooks by Martello & Toth [14] and Kellerer et al. [12] for surveys.

The aim of this paper is to investigate the use of relation-algebraic methods for dealing with multi-objective optimization problems. Relational algebra provides a powerful framework for solving various optimization problems with small programming effort [26,17]. Computer programs based on relational algebra are in particular short and easy to implement and there are several tools that are able to execute relational programs in a quite efficient way. Tools like RELVIEW [5] or CROCOPAT [7] represent relations implicitly by Ordered Binary Decision Diagrams (OBDDs) [3] which enables practitioners even to deal with very large relations.

The advantage of relational programs has been pointed out for many single-objective combinatorial optimization problems [4,5]. Computing an optimal solution for the considered problems often implicitly involves the consideration of the whole search space in this case. As pointed out previously, the task in multi-objective optimization is to compute a set of solutions which may in the worst case increase exponentially with the problem dimension. Using OBDDs in such cases may in particular result in a compact implicit representation of this set of solutions.

First we examine how to formulate the computation of the Pareto front for a given problem in terms of relational algebra. As this mainly relies on the intersection of quasi-orders, a relational algebraic formulation for this problem can be given in a straightforward way. The problem when computing the Pareto front in this way has to deal with the task to compute the dominance relations of the given objective functions with respect to the considered search space. As the search space is usually exponential in the input size we can only hope to be successful for problems where the relations between the different solutions are represented by OBDDs of moderate size. Later on we consider the problem of reducing the number of objectives for a given problem. Here the task is to compute a minimal subset of the given objective functions that represents the same weak dominance relation as the one implied by the set of all objectives. This problem has recently been shown to be NP-hard [8] by a reduction to the set covering problem. In the same paper an exact algorithm with worst-case exponential runtime has been proposed. We develop an algorithm on the basis of relational algebra for this problem which outperforms the one of Brockhoff & Zitzler [8] drastically in our experimental studies. The investigations show that our algorithm is able to deal with large sets of objective functions which further shows the advantage of the relation-algebraic approach.

The outline of the paper is as follows. In Sect. 2 we introduce basic preliminaries on relational algebra and multi-objective optimization. Sect. 3 gives a relation-algebraic formulation for computing the Pareto optimal search points of a given problem and Sect. 4 shows how relational algebra can be used to reduce the number of necessary objectives. The results of our experimental studies are presented in Sect. 5 and finally we finish with some conclusions.

2 Multi-objective Optimization and Relational Algebra

In this section we describe relation-algebraic preliminaries which are necessary to understand the development of algorithms. A more comprehensive presentation on the use of relational algebra can be found in [17]. Afterwards we give an introduction to the field of multi-objective optimization using the terminology of relational algebra.

2.1 Basic Principles of Relational Algebra

A concrete relation is a subset of a cartesian product $X \times Y$ of two sets. We write $R : X \leftrightarrow Y$ and denote the set of all relations of the *type* $X \leftrightarrow Y$ by $[X \leftrightarrow Y]$. In the case of finite supports, we may consider a relation as a Boolean matrix and use matrix terminology and matrix notation in the following. Especially, we speak of the rows, columns and entries of R and write R_{ij} instead of $(i, j) \in R$. In some cases, especially if the relation is an order or preorder $\leq$, we also use the infix notation $i \leq j$ to increase readability. We assume the reader to be familiar with the basic operations on relations, viz. $R^\top$ (transposition), $\overline{R}$ (negation), $R \cup S$ (union), $R \cap S$ (intersection), RS (composition), and the special relations $\mathbf{O}$ (empty relation), $\mathbf{L}$ (universal relation), and $\mathbf{I}$ (identity relation). A relation R is called *vector*, if $RL = R$ holds. As for a vector therefore the range is irrelevant, we consider in the following vectors $v : X \leftrightarrow \mathbf{1}$ with a specific singleton set $\mathbf{1} = \{\perp\}$ as range and write v_i instead of $v_{i\perp}$. Such a vector can be considered as a Boolean matrix with exactly one column, i.e. as a Boolean column vector, and describes the subset $\{x \in X : v_x\}$ of X . A vector v is called a *point* if it is injective and surjective. For $v : X \leftrightarrow \mathbf{1}$ these properties mean that it describes a singleton set, i.e. an element of X . In the matrix model, a point is a Boolean column vector in which exactly one component is true. A relation $R : X \leftrightarrow Y$ can be considered as a list of $|Y|$ vectors, the columns of R . We denote the y -th column of R with $R^{(y)}$, i.e. $R^{(y)}$ is a vector of type $X \leftrightarrow \mathbf{1}$ and for all $x \in X$ the expressions $R_x^{(y)}$ and R_{xy} are equivalent.

For all sets X and Y there exist a pair (π, ρ) of *natural projections* of $X \times Y$, i.e. two relations $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ with

$$\pi_{\langle x, y \rangle x'} \iff x = x' \quad \text{and} \quad \rho_{\langle x, y \rangle y'} \iff y = y'$$

As discussed in [17], the natural projections permit the definition of a Boolean lattice isomorphism $vec : [X \leftrightarrow Y] \rightarrow [X \times Y \leftrightarrow \mathbf{1}]$ by $vec(R) = (\pi R \cap \rho) \mathbf{L}$. With

this mapping each relation R can be represented by a vector $r = \text{vec}(R)$ in the sense that

$$r_{\langle x, y \rangle} \iff R_{xy}.$$

The inverse mapping rel is given by $\text{rel}(r) = \pi^\top(\rho \cap r\mathbf{L})$. The mapping vec allows to establish the following representation of sets of relations. A subset $\mathcal{S} = \{R_1, \dots, R_n\}$ of $[X \leftrightarrow Y]$ can be modelled by a relation $S : X \times Y \leftrightarrow [1..n]$ such that for each $i \in [1..n]$ the equation $S^{(i)} = \text{vec}(R_i)$ is satisfied, i.e., every column of S is the vector representation of a relation in $\mathcal{S}$.

2.2 Multi-objective Optimization

Many problems in computer science deal with the optimization of one single objective function which should be optimized under a given set of constraints. In this case there is a linear preorder on the set of search points and an optimal solution can be defined as a smallest (or greatest) element with respect to this preorder depending on whether we consider minimization or maximization problems. The goal is to compute exactly one smallest element with respect to the given preorder.

In the case of multi-objective optimization (see, e.g. Ehrgott [9]), several objective functions are given. These functions define a partial preference on the given set of search points. Most of the best known single-objective polynomially solvable problems like shortest path or minimum spanning tree become NP-hard when at least two weight functions have to be optimized at the same time. In this sense, multi-objective optimization is considered at least as difficult as single-objective optimization.

For multi-objective optimization problems the objective function $f = (f_1, \dots, f_k)$ is vector-valued, i.e., $f : \mathcal{S} \rightarrow \mathbb{R}^k$. Since there is no canonical complete order on $\mathbb{R}^k$, one compares the quality of search points with respect to the canonical partial order on $\mathbb{R}^k$, namely $f(x) \leq f(x')$ iff $f_i(x) \leq f_i(x')$ for all $i \in [1..k]$. A Pareto optimal search point $s \in \mathcal{S}$ is a point such that (in the case of minimization problems) $f(x)$ is minimal with respect to this partial order and all $f(s')$, $s' \in \mathcal{S}$. In terms of relational algebra the problem can be stated as follows.

Definition 1. *Given a minimization problem in a search space $\mathcal{S}$ and a set $\mathcal{F} = \{f_1, \dots, f_k\}$ of functions $f_i : \mathcal{S} \rightarrow \mathbb{R}$, we define a set $\mathcal{R} = \{\preceq_1, \dots, \preceq_k\}$ of k relations of type $\mathcal{S} \leftrightarrow \mathcal{S}$ by $x \preceq_i x' \iff f_i(x) \leq f_i(x')$. The weak dominance relation $\preceq : \mathcal{S} \leftrightarrow \mathcal{S}$ is defined by $x \preceq x' \iff \forall i \in [1..k] : f_i(x) \leq f_i(x')$. The strong dominance relation $\prec$ is defined by $x \prec x' \iff x \preceq x' \wedge \exists i \in [1..k] : f_i(x) < f_i(x')$. We say that x dominates x' , if $x \prec x'$ holds. A search point x is called Pareto optimal if there exist no search point x' that dominates x .*

Again there can be many Pareto optimal search points but they do not necessarily have the same objective vector. The Pareto front, denoted by F , consists of all objective vectors $y = (y_1, \dots, y_k)$ such that there exists a search point s where $f(s) = y$ and $f(s') \leq f(s)$ implies $f(s') = f(s)$ for all $s' \in \mathcal{S}$. The Pareto set consists of all solutions whose objective vector belongs to the Pareto front.

The problem is to compute the Pareto front and for each element y of the Pareto front one search point s such that $f(s) = y$. We sometimes say that a search point s belongs to the Pareto front which means that its objective vector belongs to the Pareto front. The goal is to present such a set of trade-offs to a decision maker who often has to choose one single solution out of this set based on his personal preference.

Especially in the case of multi-objective optimization, evolutionary algorithms seem to be a good heuristic approach to obtain a good set of solutions. Evolutionary algorithms have the advantage that they work at each time step with a set of solutions called the population. This population is evolved to obtain a good approximation of the Pareto front. The final set of solutions presented to a decision maker should represent the different trade-offs with respect to the given objective functions. It has been pointed out in [8] that often not all objectives are necessary to represent the different trade-offs. Reducing the number of objectives that have to be examined by a decision maker may simplify the decision which of the presented solutions should be finally chosen.

3 Computing the Pareto-optimal Set

The classical problem that arises in multi-objective optimization is to compute for each objective vector belonging to the Pareto front a corresponding solution of the Pareto optimal set. In the following we show how this problem can easily be solved for small problem instances where the weak dominance relation can be expressed for each function as an OBDD of moderate size.

We consider the set $\mathcal{R}$ of relations introduced in Definition 1. Every relation $\preceq_i$ in $\mathcal{R}$ is a linear preorder, i.e reflexive and transitive, and $x \preceq_i x' \vee x' \preceq_i x$ holds for each two search points x and x' . From the definition immediately follows the equation

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i$$

to describe the weak dominance relation. Hence, the relation $\preceq$, as an intersection of preorders, is also a preorder, but not necessarily linear. As discussed above, we model the set $\mathcal{R} = \{\preceq_1, \dots, \preceq_k\}$ by the relation $R : \mathcal{S} \times \mathcal{S} \leftrightarrow [1..k]$, such that $R^{(i)} = \text{vec}(\preceq_i)$ holds for each $i \in [1..k]$. In other words, each preorder $\preceq_i$ is modeled by a column of the relation R , and $R_{(x,x')i}$ is equivalent to $x \preceq_i x'$ for all search points x, x' and all $i \in [1..k]$. With this representation of the set $\mathcal{R}$ it is quite simple to compute the weak dominance relation, modeled by a vector w of type $\mathcal{S} \times \mathcal{S} \leftrightarrow \mathbf{1}$. It holds

$$w := \text{vec}(\preceq) = \overline{RL},$$

where L is the universal vector of type $[1..k] \leftrightarrow \mathbf{1}$. This equation is a special case of Theorem 1 in the next section, therefore we do not prove the equation now. We obtain the RELVIEW function

$$\text{weakDom}(R) = - (-R * L1n(R)^\sim).$$

to determine the weak dominance relation in vector representation. Given the weak dominance relation $\preceq$, the strong dominance relation $\prec$ can be computed by

$$\prec = \preceq \cap \overline{\preceq}^\top$$

because for two search points x' and x it holds

$$\begin{aligned} x' \prec x &\iff \forall i \in [1..k] : f_i(x') \leq f_i(x) \wedge \exists i \in [1..k] : f_i(x') < f_i(x) \\ &\iff x' \preceq x \wedge \neg \forall i \in [1..k] : f_i(x) \leq f_i(x') \\ &\iff x' \preceq x \wedge \neg(x \preceq x') \\ &\iff x' \preceq x \wedge x' \overline{\preceq}^\top x \\ &\iff x' (\preceq \cap \overline{\preceq}^\top) x. \end{aligned}$$

This leads to the following RELVIEW program **strongDom**, where the second parameter is an arbitrary relation of type $\mathcal{S} \leftrightarrow \mathcal{S}$ which is necessary to compute the relation representation of the weak dominance relation.

```
strongDom(R,Q)
  DECL w,W,S
  BEG w = weakDom(R);
      W = rel(w,Q);
      S = W & -W~
  RETURN S
END.
```

Based on the strong dominance relation we can compute the set of all Pareto optimal search points. An element $x \in \mathcal{S}$ is Pareto optimal if there exist no $x' \in \mathcal{S}$ with $x' \prec x$. It follows that

$$\begin{aligned} x \text{ is Pareto optimal} &\iff \neg \exists x' : x' \prec x \\ &\iff \neg \exists x' : x \prec^\top x' \\ &\iff \neg (\prec^\top \mathbf{L})_x \\ &\iff \overline{(\prec^\top \mathbf{L})}_x. \end{aligned}$$

Hence, the set of Pareto optimal search points is represented by the vector o of type $\mathcal{S} \leftrightarrow \mathbf{1}$ defined by

$$o = \overline{\prec^\top \mathbf{L}}$$

and we obtain the RELVIEW function

$$\text{ParetoOpt}(\mathbf{R}, \mathbf{Q}) = -(\text{strongDom}(\mathbf{R}, \mathbf{Q}) \wedge * \text{Ln1}(\mathbf{Q})).$$

In the set of Pareto optimal search points there can exist elements with the same fitness vector. In most cases one is interested in obtaining only one Pareto

optimal search point for each fitness vector of the Pareto front. With the equivalence relation

$$\approx := \preceq \cap \preceq^\top$$

we have $x \approx x' \iff f(x) = f(x')$ for each $x, x' \in \mathcal{S}$. Obviously, the whole equivalence class $[x]_\approx$ is Pareto optimal if x is Pareto optimal. To determine a vector $r \subseteq o$ of representatives of the equivalence classes which are Pareto optimal, we use a linear order O and adopt the smallest element of each Pareto optimal equivalence class w. r. t. O . It holds

$$\begin{aligned} r_x &\iff o_x \wedge \forall x' : x \approx x' \rightarrow O_{xx'} \\ &\iff o_x \wedge \neg \exists x' : x \approx x' \wedge \overline{O}_{xx'} \\ &\iff o_x \wedge \neg \exists x' : (\approx \cap \overline{O})_{xx'} \\ &\iff o_x \wedge \overline{(\approx \cap \overline{O})}_{L_x} \\ &\iff (o \cap \overline{(\approx \cap \overline{O})})_x. \end{aligned}$$

We obtain the vector

$$r = o \cap \overline{(\approx \cap \overline{O})}_L$$

which contains exactly one representative of each Pareto optimal equivalence class with the following RELVIEW program, where O is a linear order.

```
ParetoOptRep(R,O)
DECL W,o,r
BEG W = rel(weakdom(R),O);
    o = ParetoOpt(R,O);
    r = o & -( (W & W~ & -O)*L(o) )
RETURN r
END.
```

4 Reducing the Number of Objectives

Often multi-objective problems involve a large set of objectives for which the task is to compute a good approximation of the Pareto front. Often not all objectives are necessary to describe the approximation found by running some heuristic method such as an evolutionary algorithm [8]. In this case we are faced with the problem of computing a cardinality-wise minimal subset of objectives that preserves the same preference relation of the original set of objectives. Dealing with such a smaller set of objectives may make the decision easier for a decision maker which of the possible alternatives finally to choose.

In the following we deal with a given subset $X \subseteq \mathcal{S}$ instead of the whole search space. Therefore, we assume the introduced preorders $\preceq_1, \dots, \preceq_k$ and $\preceq$ as relations of type $X \leftrightarrow X$. We consider the MINIMUM OBJECTIVE SUBSET PROBLEM introduced in [8] which can be defined as follows.

Definition 2. MINIMUM OBJECTIVE SUBSET PROBLEM

Given a set of solutions, the weak Pareto dominance relation $\preceq$ and for all objective functions $f_i \in \mathcal{F}$ the single relations $\preceq_i$ where

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i .$$

Compute a subset $T \subseteq [1..k]$ of minimum size with

$$\preceq = \bigcap_{i \in T} \preceq_i .$$

As described in Sect. 2.2, we model the set $\mathcal{R} = \{\preceq_1, \dots, \preceq_k\}$ by a relation $R : X \times X \leftrightarrow [1..k]$. Based on this relation R and the representation of subsets of $[1..k]$ by vectors $[1..k] \leftrightarrow \mathbf{1}$ (see Sect. 2.1) the following theorem states a relational expression to describe intersections of subsets of $\mathcal{R}$.

Theorem 1. For every subset $T \subseteq [1..k]$ it holds

$$\text{vec}\left(\bigcap_{i \in T} \preceq_i\right) = \overline{Rt},$$

where t is the vector of type $[1..k] \leftrightarrow \mathbf{1}$ that models the set T .

Proof. Using the definition of R and the fact that vec is a lattice isomorphism, we obtain $\text{vec}(\bigcap_{i \in T} \preceq_i) = \bigcap_{i \in T} \text{vec}(\preceq_i) = \bigcap_{i \in T} R^{(i)}$. For $y = \langle x, x' \rangle \in X \times X$ it follows

$$\begin{aligned} \text{vec}\left(\bigcap_{i \in T} \preceq_i\right)_y &\iff \left(\bigcap_{i \in T} R^{(i)}\right)_y \\ &\iff \forall i \in T : R_y^{(i)} \\ &\iff \forall i \in T : R_{yi} \\ &\iff \forall i : t_i \rightarrow R_{yi} \\ &\iff \neg \exists i : t_i \wedge \overline{R}_{yi} \\ &\iff \overline{Rt}_y. \end{aligned}$$

□

As an immediate consequence, with the set $[1..k]$ modeled by the universal vector $\mathbf{L} : [1..k] \leftrightarrow \mathbf{1}$, we obtain the vector representation of the weak dominance relation by $w = \text{vec}(\preceq) = \overline{R\mathbf{L}}$ as stated in Sect. 3.

Using the equation of Theorem 1 we can now develop a relational expression to decide if a given subset $T \subseteq [1..k]$ is feasible in the sense that the intersection $\bigcap_{i \in T} \preceq_i$ equals the weak dominance relation $\preceq$.

Theorem 2. For $T \subseteq [1..k]$ it holds

$$\preceq = \bigcap_{i \in T} \preceq_i \iff \overline{\overline{L(\overline{Rt} \cup w)}} = \mathbf{L}.$$

Proof. For every subset $T \subseteq [1..k]$ it holds $\preceq \subseteq \bigcap_{i \in T} \preceq_i$. Using Theorem 11 we obtain

$$\begin{aligned}
 \preceq &= \bigcap_{i \in T} \preceq_i \iff \bigcap_{i \in T} \preceq_i \subseteq \preceq \\
 &\iff \text{vec}\left(\bigcap_{i \in T} \preceq_i\right) \subseteq \text{vec}(\preceq) \\
 &\iff \overline{\overline{Rt}} \subseteq w \\
 &\iff \overline{\overline{Rt}} \cap \overline{w} = \mathbf{O} \\
 &\iff \overline{\overline{Rt \cup w}} = \mathbf{O} \\
 &\iff \mathbf{L}(\overline{\overline{Rt \cup w}}) = \mathbf{O} \\
 &\iff \overline{\overline{\mathbf{L}(\overline{\overline{Rt \cup w}})}} = \mathbf{L}.
 \end{aligned}
 \quad \square$$

Theorem 12 leads to a mapping $\varphi_{\text{cut}} : [[1..k] \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$ defined by

$$\varphi_{\text{cut}}(t) = \overline{\overline{\mathbf{L}(\overline{\overline{Rt \cup w}})}}$$

to test if the vector t models a suitable subset to reduce the number of objectives, i.e. it holds

$$\varphi_{\text{cut}}(t) = \mathbf{L} \iff \preceq = \bigcap \{ \preceq_i \mid t_i \}.$$

Since φ_{cut} is a vector predicate in the sense of 11, it can be generalized to a test-mapping φ_{cut}^Z for evaluating the columns of relations of type $[1..k] \leftrightarrow Z$. More formally, we obtain for every set Z a mapping $\varphi_{\text{cut}}^Z : [[1..k] \leftrightarrow Z] \rightarrow [\mathbf{1} \leftrightarrow Z]$ by defining

$$\varphi_{\text{cut}}^Z(M) = \overline{\overline{\mathbf{L}(\overline{\overline{RM \cup w\mathbf{L}}})}},$$

where $\mathbf{L}$ is the universal relation of type $\mathbf{1} \leftrightarrow Z$. For every relation $M : [1..k] \leftrightarrow Z$, the row vector $\varphi_{\text{cut}}^Z(M)$ represents the columns of M which model the subsets of $[1..k]$ which can be used to reduce the number of objectives, i.e it holds

$$\varphi_{\text{cut}}^Z(M)_{\perp j} \iff \varphi_{\text{cut}}(M^{(j)}) = \mathbf{L} \iff \preceq = \bigcap \{ \preceq_i \mid M_i^{(j)} \}.$$

By applying this approach to the membership relation $\mathbf{M} : [1..k] \leftrightarrow 2^{[1..k]}$ which models the power set of $[1..k]$, we are able to compute all suitable subsets. $\mathbf{M}$ is defined by

$$\mathbf{M}_{xY} \iff x \in Y$$

and lists all subsets of $[1..k]$ columnwise. With $\varphi_{\text{cut}}^{2^{[1..k]}}(\mathbf{M})$ we obtain a row vector $c : \mathbf{1} \leftrightarrow 2^{[1..k]}$ that specifies all subsets $T \subseteq [1..k]$ with $\preceq = \bigcap_{i \in T} \preceq_i$.

The test mapping $\varphi_{\text{cut}}^{2^{[1..k]}}$ leads to the following RELVIEW program where `epsi(L1n(R^))` generates the membership relation of type $[1..k] \leftrightarrow 2^{[1..k]}$.

```

cut(R)
  DECL w, M, c
  BEG w = weakDom(R);
      M = epsi(L1n(R^));
      c = -(L1n(R)^ * -(R * M | w * L1n(M)))
  RETURN c
END.

```

The next step is to find the smallest subsets with this property. To this end, we use the size-comparison relation $C : 2^{[1..k]} \leftrightarrow 2^{[1..k]}$, defined by $C_{AB} \iff |A| \leq |B|$ and define a mapping se , which computes for a given linear preorder relation Q and a vector v the smallest elements in v w.r.t. Q . More formally, with $se(Q, v) = v \cap \overline{Q}v$ we obtain a vector such that

$$se(Q, v)_x \iff v_x \wedge \forall y : v_y \rightarrow Q_{xy}$$

holds. The immediate consequence is the following RELVIEW function **se** to compute smallest elements.

$$se(Q, v) = v \ \& \ -(Q * v).$$

With $s = se(C, c^\top)$ we obtain all subsets $T \subseteq [1..k]$ with the smallest cardinality that satisfy the property $\preceq = \bigcap_{i \in T} \preceq_i$. More formally, s is a vector of type $2^{[1..k]} \leftrightarrow 1$ with $s \subseteq c^\top$ and it holds

$$s_j \iff \preceq = \bigcap \{ \preceq_i \mid M_i^{(j)} \} \wedge \forall \ell : |M^{(\ell)}| < |M^{(j)}| \rightarrow \preceq \neq \bigcap \{ \preceq_i \mid M_i^{(\ell)} \}.$$

Hence each entry of s specifies a column of M that represents a suitable subset of $[1..k]$ with the smallest cardinality. By using the vector predicate φ_{cut} we can express the equivalence above as follows.

$$s_j \iff \varphi_{cut}(M^{(j)}) = L \wedge \forall \ell : |M^{(\ell)}| < |M^{(j)}| \rightarrow \varphi_{cut}(M^{(\ell)}) = O$$

The following RELVIEW program computes the vector s . The size comparison relation on the power set $2^{[1..k]}$ is generated by **cardrel**(**L1n**(**R**)[^]) .

```

smallCuts(R)
  DECL c, C, s
  BEG c = cut(R);
      C = cardrel(L1n(R)^);
      s = se(C, c^);
  RETURN s
END.

```

5 Experimental Results

In this section, we present the experimental results obtained for the objective reduction approach described in the previous section. We have carried out all of

$$S =$$

	1	2	3	4	5
1		■			
2			■		
3				■	
4					■
5					

Fig. 1. The 5×5 successor relation

these computations using the RELVIEW system which permits the evaluation of relation-algebraic terms and programs. All our computations were executed on a Sun Blade 1500 running Solaris 9 at 1000 MHz.

5.1 Results for Random Preorders

We have tested our program with instances of up to 145 randomly generated preorders computed by the RELVIEW system. Generating a random total order relation of the type $X \leftrightarrow X$ is rather simple. Based on a given total hasse relation S and a randomly generated permutation P , both of the type $X \leftrightarrow X$, we obtain a random linear order by $O = (PSP^\top)^*$, the reflexive-transitive closure of the hasse relation $PSP^\top$.

The following RELVIEW program generates a random total order in this way, where the input Q is an arbitrary relation of type $X \leftrightarrow X$, `succ(Ln1(Q))` gives the successor relation (see Fig. 1 as an example) of the same type and `randomperm(Ln1(Q))` computes a random permutation.

```

randomOrder(Q)
  DECL S,P,O
  BEG S = succ(Ln1(Q));
      P = randomperm(Ln1(Q));
      O = refl(trans(P*S*P^))
  RETURN O
END.
```

To obtain a preorder, we have to include some additional entries in the random order relation. To this end, we generate a random relation A and add $A \cup A^\top$ to $PSP^\top$ before computing the reflexive-transitive closure. Hence, the preorder is given by $(PSP^\top \cup A \cup A^\top)^*$. We use $A \cup A^\top$ instead of A to ensure that we get new entries which are not contained in the order relation $(PSP^\top)^*$ and therefore obtain a preorder instead of an order relation. The following RELVIEW program generates a random preorder in this way, where the input is a nonempty relation which determines the type and influences the number of entries of the determined preorder. With `random(Q,Q)` a random relation $A : X \leftrightarrow X$ is generated such that for all $i, j \in X$ the probability of A_{ij} being true is $|Q|/|X|^2$.

```

randomPreOrder(Q)
  DECL S,P,A,Pre0
  BEG S = succ(Ln1(Q));
      P = randomperm(Ln1(Q));
      A = random(Q,Q);
      Pre0 = refl(trans(P*S*P^ | A | A^))
  RETURN Pre0
END.

```

Using this program it is simple to produce random inputs consisting of k randomly generated preorders, modelled as a relation $R : X \times X \leftrightarrow [1..k]$. The following program successively determines k preorders **Pre0** of type $X \leftrightarrow X$ and their vector representation **pre0**. With $R = R \mid \text{pre0} * \mathbf{p}^{\wedge}$, where $\mathbf{p}$ is a point representing an element $i \in [1..k]$, the vector **pre0** is inserted into R as the i -th column.

```

randomInput(Q,k)
  DECL R,z,Pre0,pre0,p
  BEG R = 0(vec(Q)*k^);
      z = k
      WHILE -empty(z) DO
        Pre0 = randomPreOrder(Q);
        pre0 = vec(Pre0);
        p = point(z);
        R = R | pre0*p^;
        z = z & -p
      OD
  RETURN R
END.

```

Our experimental results with respect to random preorders are given in Tab. [11](#). Depending on the probability used in our random function (which includes additional entries into the preorder relation) the results are shown. Note that such entries imply that solutions become indifferent which means that they have the same objective value with respect to the considered function. Tab. [11](#) shows that problems become easier with increasing this probability. The reason for that is that the number of different trade-offs becomes smaller when making solutions indifferent. Depending on the choice of this probability RELVIEW is able to deal with problems that involve 50 solutions and up to 145 objectives. The computation time for each instance is always less than 80 seconds.

5.2 Results for Knapsack Problems

A well-known problem in combinatorial optimization is the knapsack problem [\[12\]\[14\]](#) where a set of n items is given. With each item $j \in [1..n]$, a profit p_j and a weight w_j is associated. In addition a weight bound W is given and the goal is to select items such that the profit is maximized under the given weight constraint W . Omitted the weight constraints and optimizing both the profit

Table 1. Results for random preorders with different values of p , where runtimes are given in seconds and the respective second columns give the reduced number of objectives

p	1/2500		1/500		1/250		3/500		1/125		1/50	
# obj	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.
5	0.04	5	0.01	5	0.01	5	0.01	4	0.01	4	0.16	2
15	0.63	8	0.51	15	0.48	14	0.46	12	0.47	9	0.38	3
25	5.71	7	0.25	11	0.07	14	0.07	14	0.06	14	0.05	8
35			1.22	13	0.20	22	0.16	23	0.16	14	0.15	5
45					0.49	26	0.41	28	0.41	25	0.38	19
55					39.06	17	0.87	30	0.80	30	0.79	18
65					5.29	23	2.44	29	3.19	32	1.54	17
75							2.91	34	2.72	38	3.52	23
85							7.86	27	5.61	33	4.43	20
95									9.24	40	7.87	27
105									11.60	40	23.86	21
115									16.80	42	24.43	26
125									27.37	44	23.19	29
135											32.52	32
145											77.77	30

Table 2. Comparison of the relational approach with the exact one given in [8] where runtimes are given in milliseconds

Objectives	Runtime RELVIEW	Runtime Exact Approach [8]
5	40	178
10	70	4369
15	590	166343
20	170	197690
25	430	5135040
30	1360	3203227
35	5990	—

and the weight simultaneously, Beier & Vöcking [1] have shown that for various input distributions the size of the Pareto front is polynomially bounded in the number of items. Their results imply that the well-known dynamic programming approach due to Nemhauser & Ullman [15] is able to enumerate these solutions in expected polynomial time. In the multi-objective knapsack problem [20], k knapsack problems are considered simultaneously. In this case we are faced with k knapsacks where knapsack i has capacity W_i . The weight of item j in knapsack i is denoted by w_{ij} and its profit by p_{ij} . The goal is to maximize for each knapsack i the function $f_i(x) = \sum_{j=1}^n p_{ij}x_j$ such that $w_i(x) = \sum_{j=1}^n w_{ij}x_j \leq W_i$ holds. Hence, the problem is given by the function $f = (f_1, \dots, f_k)$ which should be optimized under the different weight constraints of the k knapsacks.

We also investigated this problem in the same setting as done in [8]. The different solutions on which the objective reduction algorithms are executed are computed by running a multi-objective evolutionary algorithm called SPEA2 [19] on random instances with different number of objective functions. To compare the relation- algebraic approach with respect to efficiency we used the implementation of Brockhoff & Zitzler [8]. The results are given in Tab. 2 and show that the RELVIEW program outperforms the previous approach drastically. RELVIEW is able to compute for each instance an optimal solution within 6 seconds while the approach of Brockhoff and Zitzler needs large computation times and is unable to deal with instances which have more than 30 objectives.

6 Conclusions

In contrast to single-objective problems where one single optimal solution should be computed, the aim in multi-objective optimization is to compute solutions that represent the different trade-offs with respect to the objective functions. We have done a first step in examining such problems in terms of relational algebra and considered two important issues when dealing with multi-objective optimization. For the classical problem of computing the Pareto optimal solutions we have given a relation-algebraic approach that leads to a short RELVIEW program which is at least able to deal with instances of moderate size. We have also examined the problem of reducing the number of objectives to be presented to a decision maker. It turns out that the relation-algebraic approach is very efficient for this problem and can deal with a large number of objectives. The comparison for the multi-objective knapsack problem shows that our algorithm outperforms the previous one drastically.

Acknowledgement

We thank Dimo Brockhoff and Eckart Zitzler for providing the implementation of their algorithms and the test instances for the multi- objective knapsack problem.

References

1. Beier, R., Vöcking, B.: Random knapsack in expected polynomial time. *J. Comput. Syst. Sci.* 69(3), 306–329 (2004)
2. Berghammer, R.: Solving algorithmic problems on orders and lattices by relation algebra and RelView. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) *CASC 2006*. LNCS, vol. 4194, pp. 49–63. Springer, Heidelberg (2006)
3. Berghammer, R., Leoniyuk, B., Milanese, U.: Implementation of relational algebra using binary decision diagrams. In: de Swart, H. (ed.) *RelMiCS 2001*. LNCS, vol. 2561, pp. 241–257. Springer, Heidelberg (2002)
4. Berghammer, R., Milanese, U.: Relational approach to boolean logic problems. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) *RelMiCS 2005*. LNCS, vol. 3929, pp. 48–59. Springer, Heidelberg (2006)

5. Berghammer, R., Neumann, F.: RelView – an OBDD-based computer algebra system for relations. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2005. LNCS, vol. 3718, pp. 40–51. Springer, Heidelberg (2005)
6. Berghammer, R., Rusinowska, A., de Swart, H.C.M.: Applying relational algebra and RelView to coalition formation. *European Journal of Operational Research* 178(2), 530–542 (2007)
7. Beyer, D., Noack, A., Lewerentz, C.: Efficient relational calculation for software analysis. *IEEE Transactions on Software Engineering* 31(2), 137–149 (2005)
8. Brockhoff, D., Zitzler, E.: Dimensionality reduction in multiobjective optimization: The minimum objective subset problem. In: Waldmann, K.H., Stocker, U.M. (eds.) *Operations Research Proceedings 2006*, pp. 423–430. Springer, Heidelberg (2007)
9. Ehrgott, M.: *Multicriteria Optimization*, 2nd edn. Springer, Berlin (2005)
10. Goemans, M.X.: Minimum bounded degree spanning trees. In: *Proc. of FOCS 2006*, pp. 273–282. IEEE Computer Society Press, Los Alamitos (2006)
11. Kehden, B.: Evaluating sets of search points using relational algebra. In: Schmidt, R.A. (ed.) *RelMiCS/AKA 2006*. LNCS, vol. 4136, pp. 266–280. Springer, Heidelberg (2006)
12. Kellerer, H., Pferschy, U., Pisinger, D.: *Knapsack Problems*. Springer, Heidelberg (2004)
13. Könemann, J., Ravi, R.: Primal-dual meets local search: Approximating MSTs with nonuniform degree bounds. *SIAM J. Comput.* 34(3), 763–773 (2005)
14. Martello, S., Toth, P.: *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester (1990)
15. Nemhauser, G., Ullmann, Z.: Discrete dynamic programming and capital allocation. *Management Sci.* 15(9), 494–505 (1969)
16. Ravi, R., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Many birds with one stone: multi-objective approximation algorithms. In: *Proc. of STOC 1993*, pp. 438–447 (1993)
17. Schmidt, G., Ströhlein, T.: *Relations and Graphs – Discrete Mathematics for Computer Scientists*. Springer, Heidelberg (1993)
18. Singh, M., Lau, L.C.: Approximating minimum bounded degree spanning trees to within one of optimal. In: *Proc. of STOC 2007*, pp. 661–670. ACM Press, New York (2007)
19. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K.C., et al. (eds.) *Proc. of EUROGEN 2001*, pp. 95–100. CIMNE (2002)
20. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 3(4), 257–271 (1999)

The Lattice of Contact Relations on a Boolean Algebra

Ivo Düntsch and Michael Winter*

Department of Computer Science,
Brock University,
St. Catharines, Ontario, Canada, L2S 3A1
{duentsch,mwinter}@brocku.ca

Abstract. Contact relations on an algebra have been studied since the early part of the previous century, and have recently become a powerful tool in several areas of artificial intelligence, in particular, qualitative spatial reasoning and ontology building. In this paper we investigate the structure of the set of all contact relations on a Boolean algebra.

1 Introduction

Contact relations arise historically in two different contexts: Proximity relations were introduced by Efremovič to express the fact that two objects are – in some sense – close to each other [1]. The other source of contact relations is pointless geometry (or topology), which goes back to the works of [2], [3], [4], [5] and others. The main difference to traditional geometry is the way in which the building blocks are defined: Instead of taking points as the basic entity and defining other geometrical objects from these, the pointless approach starts from certain collections of points, for example, plane regions or solids, and defines points from these. One reason behind this approach is the fact that points are (unobservable) abstract objects, while regions or solids occur naturally in physical reality, as we sometimes painfully observe.

A standard example of a contact relation is the following: Consider the set of all closed disks in the plane, and say that two such disks are in contact if they have a nonempty intersection. More generally we say that two regular closed sets are in contact if they have a nonempty intersection. This relation is, indeed, considered to be the *standard contact* between regular closed sets of a topological space. Motivated by certain problems arising in qualitative spatial reasoning, Boolean algebras equipped with a contact relation have been intensively studied in the artificial intelligence community, and we invite the reader to consult [6] or [7] for some background reading.

2 Notation and Basic Definitions

We assume that the reader has a working knowledge of lattice theory, Boolean algebras, and topology. Our standard references for these are, respectively, [8], [9], and [10].

For any set U , we denote by $\text{Rel}(U)$ the set of all binary relations on U , and by $1'$ the identity relation on U . If $x \in U$, then $\text{dom}_R(x) = \{y : yRx\}$, and, if $M \subseteq U$, we let

* Both authors gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada.

$\text{dom}_R(M) = \bigcup_{x \in M} \text{dom}_R(x)$. Similarly, we define $\text{ran}_R(x)$ and $\text{ran}_R(M)$. If R is understood, we will usually drop the subscript; furthermore, we will usually write $R(x)$ for $\text{ran}_R x$.

Two distinct elements $x, z \in U$ are called *R-connected*, if there are $y_0, \dots, y_k \in U$ such that $x = y_0, z = y_k$, and $y_0 R y_1 R \dots R y_k$. If x and z are *R-connected*, we write $x \xrightarrow{R} z$. A subset W of U is called *R-connected*, if any two different elements of W are connected. A maximally *R-connected* subset of U is called a *component of R*. A *clique of R* is a nonempty subset M of U with $M \times M \subseteq R$.

Throughout, $\langle B, +, \cdot, *, 0, 1 \rangle$ will denote a Boolean algebra (BA), and $\mathbf{2}$ is the two element BA. If A is a subalgebra of B , we will write $A \leq B$. For $M \subseteq B$, $[M]$ is the subalgebra of B generated by M , and $M^+ = M \setminus \{0\}$, $M^- = M \setminus \{1\}$. If I, J are ideals of B , then $I \vee J$ denotes the ideal generated by $I \cup J$, i.e. $I \vee J = \{a : (\exists b, c)[a \in I, b \in J \text{ and } a = b + c]\}$.

$\text{At}(B)$ is the set of atoms of B , and $\text{Ult}(B)$ its set of ultrafilters. We assume that $\text{Ult}(B)$ is equipped with the Stone topology $\tau_{\text{Ult}(B)}$ via the mapping $h : B \rightarrow 2^{\text{Ult}(B)}$ with $h(x) = \{U \in \text{Ult}(B) : x \in U\}$; the product topology on $\text{Ult}(B)^2$ is denoted by $\tau_{\text{Ult}(B)^2}$. Note that $\tau_{\text{Ult}(B)^2}$ is the Stone space of the free product $B_0 \oplus B_1$, where $B_0, B_1 \cong B$, see e.g. Section 11.1. of [9].

Recall the following result for topological spaces X_0, X_1 ,

Lemma 1. [10] *Proposition 2.3.1.] If S_i is a basis for X_i , $i \leq 1$, then $\{\langle W_0, W_1 \rangle : W_0 \in S_0, W_1 \in S_1\}$ is a basis for the product topology on $X_0 \times X_1$.*

In particular, the sets of the form $h(a) \times h(b)$ where $a, b \in B$ are a basis for the product topology on $\text{Ult}(B)^2$. Furthermore, note that for $M \subseteq \text{Ult}(B)$, $F \in \text{cl}(M)$ if and only if $F \subseteq \bigcup M$.

We denote by $\text{Rel}^{rs}(\text{Ult}(B))$ the collection of all reflexive and symmetric relations on $\text{Ult}(B)$, and by $\text{Rel}^{rsc}(\text{Ult}(B))$ the collection of all reflexive and symmetric relations on $\text{Ult}(B)$ that are closed in $\tau_{\text{Ult}(B)^2}$. Note that $1' \in \text{Rel}^{rsc}(\text{Ult}(B))$, and that $\text{int}(1') \neq \emptyset$ if and only if B has an atom.

B is called a *finite-cofinite algebra* (FC-algebra), if every element $\neq 0, 1$ is a finite sum of atoms or the complement of such an element. If B is an FC-algebra, and $|B| = \kappa$, then B is isomorphic to the BA $FC(\kappa)$ which is generated by the finite subsets of κ . If $\gamma \in \kappa$, we let F_γ be the ultrafilter of $FC(\kappa)$ generated by $\{\gamma\}$, and F_κ be the ultrafilter of cofinite sets. If $M \subseteq \text{Ult}(B)$, $x \in B$, we say that M *admits* x , if $x \in \bigcap M$, i.e. if $M \subseteq h(x)$.

3 Boolean Contact Algebras

Suppose that $C \in \text{Rel}(B)$, and consider the following properties: For all $x, y, z \in B$

$$C_0. 0(-C)x$$

$$C_1. x \neq 0 \Rightarrow xCx$$

$$C_2. xCy \Rightarrow yCx$$

$$C_3. xCy \text{ and } y \leq z \Rightarrow xCz.$$

(The compatibility axiom)

$$C_4. xC(y + z) \Rightarrow (xCy \text{ or } xCz)$$

(The sum axiom)

$$C_5. C(x) = C(y) \Rightarrow x = y.$$

(The extensionality axiom)

- $C_6. (xCz \text{ or } yCz^*) \Rightarrow xCy$ (The interpolation axiom)
 $C_7. (x \neq 0 \wedge x \neq 1) \Rightarrow xCx^*$ (The connection axiom)

C is called a *contact relation* (CR), and the structure $\langle B, C \rangle$ is called a *Boolean contact algebra* (BCA), if C satisfies $C_0 - C_4$. C is called an *extensional contact relation* (ECR) if it satisfies $C_0 - C_5$. If C satisfies C_7 we call it *connected*. The collection of contact relations on B will be denoted by $\mathcal{C}_B$.

As mentioned in the introduction, a standard example for a BCA, indeed, the original motivation for studying contact relations, is the collection of regular closed sets of the Euclidean plane with *standard contact* defined by $aCb \iff a \cap b \neq \emptyset$; an in-depth investigation of BCAs in relation to topological properties can be found in [11].

Another important example of a contact relation on B is the *overlap relation* O on B defined by $xOy \iff x \cdot y \neq 0$.

Lemma 2. *C is an extensional contact relation if and only if for all $x, y \notin \mathbf{2}$ with $x \cdot y = 0$, there is some $z \in B^+$ such that $z \leq y$ and $x(-C)z$.*

Proof. “ $\Rightarrow$ ”: We have shown in [12] that for an extensional contact relation and all $z \neq 0$, $z = \sum \{t : t(-C)z^*\}$. Suppose that $x, y \notin \mathbf{2}$, and that $x \cdot y = 0$. Assume that xCz for all $0 \leq z \leq y$; then $x(-C)z$ implies that $z \cdot y = 0$, i.e. $z \leq y^*$. Since $x^* = \sum \{t : t(-C)x\}$, it follows that $x^* \leq y^*$, i.e. $y \leq x$. This contradicts the hypothesis that $y \neq 0$ and $x \cdot y = 0$.

“ $\Leftarrow$ ”: This is obvious. $\square$

The following concepts have their origin in proximity theory [1], which has a close connection to the theory of contact relations, see e.g. [13]. A *clan* is a subset Γ of B which satisfies

- Γ_1 . If $x, y \in \Gamma$ then xCy .
- Γ_2 . If $x + y \in \Gamma$ then $x \in \Gamma$ or $y \in \Gamma$.
- Γ_3 . If $x \in \Gamma$ and $x \leq y$, then $y \in \Gamma$.

In the sequel, we will use upper case Greek letters Γ, Δ etc to denote clans. When C is understood, the set of clans of $\langle B, C \rangle$ will be denoted by $\text{Clan}(B)$; clearly, each clan is contained in a maximal clan, and we will denote the set of maximal clans by $\text{MaxClan}(B)$. A *cluster* is a clan Γ for which $\{x\} \times \Gamma \subseteq C$ implies $x \in \Gamma$ for all $x \in B$.

For later use we note the following:

Lemma 3. [12] *Suppose that C is a contact relation on B . Then,*

1. aCb if and only if there is a clan containing a and b if and only if there are ultrafilters F, G of B such that $a \in F$, $b \in G$ and $F \times G \subseteq C$.
2. If $\Gamma \in \text{Clan}(B)$, then $B \setminus \Gamma$ is an ideal of B .

4 Contact Relations and Ultrafilters

The connection between (ultra-) filters on B and contact relations was established in [14], and, more generally, in [11]. Our aim in this Section is to establish the following representation theorem¹:

¹ One of the referees has kindly pointed out that a more general result has independently been shown in [15].

Theorem 1. *Suppose that B is a Boolean algebra. Then, there is a bijective order preserving correspondence between the contact relations on B and the reflexive and symmetric relations on $\text{Ult}(B)$ that are closed in the product topology of $\text{Ult}(B)^2$.*

Proof. Let $q : \text{Rel}^{\text{rsc}}(\text{Ult}(B)) \rightarrow \text{Rel}(B)$ be defined by $q(R) := \bigcup \{F \times G : \langle F, G \rangle \in R\}$; then, clearly, q preserves $\subseteq$. We first show that $q(R) \in \mathcal{C}_B$; this was shown mutatis mutandi in [14] for proximity structures, and for completeness, we repeat the proof. Since no ultrafilter of B contains 0, $q(R)$ satisfies C_0 . The reflexivity of R implies C_1 , and the symmetry of R implies C_2 . Since ultrafilters are closed under $\leq$, $q(R)$ satisfies C_3 . For C_4 , let $a \in q(R)$ ($b + c$); then, there are $F, G \in \text{Ult}(B)$ such that $a \in F$, $b + c \in G$, and $\langle F, G \rangle \in R$. Since G is an ultrafilter, $b \in G$ or $c \in G$, and it follows that aCb or aCc .

To show that q is injective, suppose that $R, R' \in \text{Rel}^{\text{rsc}}(\text{Ult}(B))$, $q(R) = q(R')$, and assume that $\langle F, G \rangle \in R' \setminus R$. Since R is closed, there are $a, b \in B$ such that $a \in F$, $b \in G$, and $(h(a) \times h(b)) \cap R = \emptyset$. Now, since $q(R) = q(R')$ it follows that $F \times G \subseteq \bigcup \{F' \times G' : \langle F', G' \rangle \in R\}$, and thus, there are $F', G' \in \text{Ult}(B)$ such that $a \in F'$, $b \in G'$, and $\langle F', G' \rangle \in R$. This contradicts $(h(a) \times h(b)) \cap R = \emptyset$.

For surjectivity, let $C \in \mathcal{C}_B$, and set $p(C) := \{\langle F, G \rangle : F \times G \subseteq C\}$. We first show that $p(C) \in \text{Rel}^{\text{rsc}}(\text{Ult}(B))$: It is straightforward to show that symmetry of C implies symmetry of $p(C)$, and C_1 implies that $p(C)$ is symmetric [14].

Next, suppose that $\langle F, G \rangle \in \text{cl}(p(C))$, and assume that $\langle F, G \rangle \notin p(C)$. Then, $F \times G \not\subseteq C$, and thus, there are $a \in F$, $b \in G$ such that $a(-C)b$. Now, $h(a) \times h(b)$ is an open neighbourhood of $\langle F, G \rangle$, and $\langle F, G \rangle \in \text{cl}(p(C))$ implies that there is some $\langle F', G' \rangle \in p(C)$ such that $\langle F', G' \rangle \in h(a) \times h(b)$. But then, $F' \times G' \subseteq C$ and $\langle a, b \rangle \in F' \times G'$ implies aCb , a contradiction.

All that remains to show is $C = q(p(C))$: By Lemma 3 and the definitions of the mappings,

$$\begin{aligned} aCb &\iff (\exists F, G)[a \in F, b \in G \text{ and } F \times G \subseteq C] \\ &\iff (\exists F, G)[a \in F, b \in G \text{ and } \langle F, G \rangle \in p(C)] \iff \langle a, b \rangle \in q(p(C)). \end{aligned}$$

This completes the proof. $\square$

Finally, we turn to the connection between clans and closed sets of ultrafilters; if $M \subseteq \text{Ult}(B)$, we let $\Gamma_M = \bigcup M$; conversely, if $\Gamma \in \text{Clan}(B)$, we set $\text{uf}(\Gamma) = \{F \in \text{Ult}(B) : F \subseteq \Gamma\}$. We will also write R_C instead of $q^{-1}(C)$.

Theorem 2. 1. $\bigcup \text{uf}(\Gamma) = \Gamma$ for each clan Γ .

2. If $\Gamma \in \text{Clan}(B)$, then $\text{uf}(\Gamma)$ is a closed clique in R_C .

3. If M is a clique in R_C , then Γ_M is a clan, and $\text{uf}(\Gamma) = \text{cl}(M)$.

4. A maximal clique M of R_C is closed.

Proof. 1. Suppose that $\Gamma \in \text{Clan}(B)$. Then,

$$\begin{aligned} x \in \bigcup \text{uf}(\Gamma) &\iff (\exists F \in \text{Ult}(B))[F \subseteq \text{uf}(\Gamma) \text{ and } x \in F], \\ &\iff (\exists F \in \text{Ult}(B))[F \subseteq \Gamma \text{ and } x \in F], \\ &\iff x \in \Gamma, \end{aligned}$$

since Γ is a union of ultrafilters.

2. It was shown in [11] that $\Gamma \in \text{Clan}(B)$ is a clique; for completeness, we give a proof:

$$\begin{aligned} \Gamma \in \text{Clan}(B) &\Rightarrow (\forall F, G \in \text{Ult}(B))[F, G \subseteq \Gamma \Rightarrow F \times G \subseteq C], \\ &\Rightarrow (\forall F, G \in \text{Ult}(B))[F, G \in \text{uf}(\Gamma) \Rightarrow F \times G \subseteq C], \\ &\Rightarrow (\forall F, G \in \text{Ult}(B))[F, G \in \text{uf}(\Gamma) \Rightarrow \langle F, G \rangle \in R_C]. \end{aligned}$$

All that remains to be shown is that $\text{uf}(\Gamma)$ is closed:

$$F \in \text{cl}(\text{uf}(\Gamma)) \iff F \subseteq \bigcup U_\Gamma \iff F \subseteq \Gamma \iff F \in \text{uf}(\Gamma).$$

3. Since Γ_M is a union of ultrafilters, it clearly satisfies [L2] and [L3]. For [L1], consider

$$\begin{aligned} x, y \in \Gamma_M &\Rightarrow (\exists F, G \in \text{Ult}(B))[F, G \in M \text{ and } x \in F, y \in G], \\ &\Rightarrow (\exists F, G \in \text{Ult}(B))[\langle F, G \rangle \in R_C \text{ and } x \in F, y \in G], \\ &\Rightarrow xCy. \end{aligned}$$

For the rest, note that

$$F \in \text{uf}(\Gamma) \iff F \subseteq \Gamma_M \iff F \subseteq \bigcup M \iff F \in \text{cl}(M).$$

4. Let M be a maximal clique of R_C ; then $\Gamma_M \in \text{Clan}(B)$. By 2. above, $\text{uf}(\Gamma)$ is a closed clique that contains M . Maximality of M now implies that $M = \text{uf}(\Gamma)$, and thus, M is closed. $\square$

5 The Lattice of Contact Relations

In this section we will show that $\mathcal{C}_B$ is a lattice under the inclusion ordering. We will do this in two steps: First, we show that $\text{Rel}^{\text{rsc}}(\text{Ult}(B))$ is a lattice and then, with the help of Theorem 11, we show how to carry it over to $\mathcal{C}_B$.

It is well known that the collection $\mathcal{T}$ of closed sets of a T_1 space X is a complete and atomic dual Heyting algebra under the operations

$$\bigvee A = \text{cl}(\bigcup A), \quad \bigwedge A = \bigcap A, \quad a \stackrel{d}{\rightarrow} b = \text{cl}(b \cap -a), \quad 0 = \emptyset, \quad 1 = X, \quad (1)$$

where $A \subseteq \mathcal{T}$, and $a, b \in \mathcal{T}$. Since X is a T_1 space, the atoms of $\mathcal{T}$ are the singletons.

Theorem 3. *The collection $\text{Rel}^{\text{rsc}}(\text{Ult}(B))$ of closed reflexive and symmetric relations on $\text{Ult}(B)$ is a complete and atomic sublattice of the lattice of closed sets of $\text{Ult}(B)^2$ with smallest element $1'$, largest element is $\text{Ult}(B)^2$, and a dual Heyting algebra where $R \stackrel{d}{\rightarrow} S := \text{cl}(R \setminus S) \cup 1'$. Its atoms have the form $1' \cup \{\langle F, G \rangle, \langle G, F \rangle\}$, where F and G are distinct ultrafilters of B .*

Proof. Since $1'$ is the smallest reflexive and symmetric relation on $\text{Ult}(B)$, and closed since $\tau_{\text{Ult}(B)}$ is compact and Hausdorff, it is the smallest element of $\text{Rel}^{\text{rsc}}(\text{Ult}(B))$, and,

clearly, $\text{Ult}(B)^2$ is the largest element of $\text{Rel}^{rsc}(\text{Ult}(B))$. Since $\tau_{\text{Ult}(B)^2}$ is a T_1 space, singletons are closed, and therefore, atoms have the form $1' \cup \{\langle F, G \rangle, \langle G, F \rangle\}$ for $F, G \in \text{Ult}(B)$, $F \neq G$.

By the remarks preceding the Theorem, all that is left to show is that the operations $\vee$ and $\wedge$ do not destroy reflexivity or symmetry, and that $R \xrightarrow{d} S \in \text{Rel}^{rsc}(\text{Ult}(B))$. Let $\mathcal{R} = \{R_i : i \in I\} \subseteq \text{Rel}^{rsc}(\text{Ult}(B))$. Since the intersection of reflexive symmetric relations is a reflexive and symmetric relation, and the intersection of closed sets is closed, we have $\bigwedge \mathcal{R} = \bigcap \mathcal{R} \in \text{Rel}^{rsc}(\text{Ult}(B))$.

Set $R = \bigcup \mathcal{R}$, and observe that R is reflexive and symmetric. Let $\langle F, G \rangle \in \text{cl}(R)$, and $h(x) \times h(y)$ be a basic neighbourhood of $\langle F, G \rangle$; then $(h(x) \times h(y)) \cap R \neq \emptyset$. Since R is symmetric, $(h(y) \times h(x)) \cap R \neq \emptyset$, and, since every basic neighbourhood of $\langle G, F \rangle$ is of the form $h(y) \times h(x)$ for an open neighbourhood $h(x) \times h(y)$ of $\langle F, G \rangle$, we conclude that $\langle G, F \rangle \in \text{cl}(R)$. It follows that $\bigvee \mathcal{R} \in \text{Rel}^{rsc}(\text{Ult}(B))$.

Finally, let $R, S \in \text{Rel}^{rsc}(\text{Ult}(B))$, and $\langle F, G \rangle \in \text{cl}(R \setminus S)$. Then, $R \setminus S$ is a symmetric relation, and we have shown in the preceding paragraph that the closure of a symmetric relation is symmetric. Now, by [\[11\]](#), $\text{cl}(R \setminus S)$ is the smallest closed set T of $\tau_{\text{Ult}(B)^2}$ with $R \subseteq S \cup T$, and, since $1'$ is closed, $R \xrightarrow{d} S$ is the smallest element T of $\text{Rel}^{rsc}(\text{Ult}(B))$ with $R \subseteq S \cup T$. $\square$

Corollary 1. $\mathcal{C}_B$ is a complete and atomic dual Heyting algebra with smallest element O , largest element $B^+ \times B^+$ and the operations

$$\begin{aligned} \sum \{C_i : i \in I\} &= q \left(\bigvee_i q^{-1}(C_i) \right), \\ \prod \{C_i : i \in I\} &= q \left(\bigwedge_i q^{-1}(C_i) \right), \\ C \xrightarrow{d} C' &= q(q^{-1}(C) \xrightarrow{d} q^{-1}(C')). \end{aligned}$$

Furthermore, if $\{C_\alpha : \alpha \in I\}$ is a descending chain of contact relations, then $\bigwedge_{\alpha \in I} C_\alpha = \bigcap_{\alpha \in I} C_\alpha$.

Proof. First, recall that $aOb \iff a \cdot b \neq 0$; then, $O = \bigcup \{F \times F : F \in \text{Ult}(B)\}$, and it follows that $q(1') = O$. Clearly, $q(\text{Ult}(B) \times \text{Ult}(B)) = B^+ \times B^+$, and the atoms of $\mathcal{C}_B$ are the relations of the form $O \cup (F \times G) \cup (G \times F) = q(1' \cup \{\langle F, G \rangle, \langle G, F \rangle\})$, where $F, G \in \text{Ult}(B)$ and $F \neq G$.

Since $q : \text{Rel}^{rsc}(\text{Ult}(B)) \rightarrow \mathcal{C}_B$ is bijective and order preserving by Theorem [\[1\]](#) and $\text{Rel}^{rsc}(\text{Ult}(B))$ is a complete and atomic dual Heyting algebra, so is $\mathcal{C}_B$ with the indicated operations.

In proving the final claim, the only not completely trivial case is [\[C4\]](#). Let $a(\bigcap_{\alpha \in I} C_\alpha)$ $(s+t)$, and assume that $a(-\bigcap_{\alpha \in I} C_\alpha)s$ and $a(-\bigcap_{\alpha \in I} C_\alpha)t$. Then, there are $\alpha, \beta \in I$ such that $\alpha \leq \beta$ and $a(-C_\alpha)s$, $a(-C_\beta)t$. From $C_\beta \subseteq C_\alpha$ we obtain $a(-C_\beta)s$ and $a(-C_\beta)t$, contradicting $aC_\beta(s+t)$. $\square$

The explicit definition of the operations in $\mathcal{C}_B$ is somewhat involved, except for the supremum: Suppose that $\mathcal{R} = \{R_i : i \in I\} \subseteq \text{Rel}^{rsc}(\text{Ult}(B))$; then,

$$\begin{aligned}
 \langle a, b \rangle \in q\left(\bigvee \mathcal{R}\right) &\iff \langle a, b \rangle \in q(\text{cl}(\bigcup_{i \in I} R_i)), \\
 &\iff (\exists \langle F, G \rangle \in \text{cl}(\bigcup_{i \in I} R_i))[\langle a, b \rangle \in F \times G], \\
 &\iff (\exists \langle F_0, G_0 \rangle \in \bigcup_{i \in I} R_i)[\langle a, b \rangle \in F_0 \times G_0], \text{ since } \text{cl}(\bigcup \mathcal{R}) \text{ is closed,} \\
 &\iff (\exists i \in I)[\langle a, b \rangle \in F_0 \times G_0 \text{ and } \langle F_0, G_0 \rangle \in R_i] \\
 &\iff (\exists i \in I)[\langle a, b \rangle \in q(R_i)], \\
 &\iff \langle a, b \rangle \in \bigcup_{i \in I} q(R_i),
 \end{aligned}$$

so that supremum in $\mathcal{C}_B$ is just the union. Regarding the meet, it can be shown that

$$\prod\{C_i : i \in I\} = \{\langle a, b \rangle \in \bigcap\{C_i : i \in I\} : (\forall s, t)[b = s + t \Rightarrow x\left(\bigcap_{i \in I} C_i\right)s \text{ or } a\left(\bigcap_{i \in I} C_i\right)t]\};$$

we omit the somewhat tedious calculations. Note that the meet operation in $\mathcal{C}_B$ is usually not set intersection. For a simple example, let B be the BA with atoms a, b, c, d , and let $C_0 = O \cup (F_a \times F_b) \cup (F_b \times F_a)$, and $C_1 = O \cup (F_c \times F_d) \cup (F_d \times F_c)$. Then, $(a + c)(C_0 \cap C_1)(b + d)$, but $C_0 \cap C_1$ does not satisfy C_4 .

Since the Stone topology of a finite BA is discrete, we note

Corollary 2. *If B is finite, then $\mathcal{C}$ is isomorphic to $\text{Rel}^{rs}(\text{Ult}(B))$.*

Since the ultrafilters of a finite BA are determined by $\text{At}(B)$, the contact relations on B are uniquely determined by the reflexive and symmetric relations on $\text{At}(B)$. Thus, the adjacency relations of I16 determine the contact relations on finite BAs and vice versa.

In the sequel we shall usually write R_C (or just R , if C is understood) instead of $p(C)$ to indicate that $p(C) \in \text{Rel}(\text{Ult}(B))$. Furthermore, we let $\hat{R} = R \setminus 1'$.

Now that we have established the overall algebraic structure of $\mathcal{C}$, we consider collections of contact relations on B that satisfy additional axioms; for $5 \leq i \leq 7$, set $\mathcal{C}_i = \{C \in \mathcal{C} : C \models C_i\}$.

If $B \neq \mathbf{2}$, then for the bounds of $\mathcal{C}$ we observe

$$O \in \mathcal{C}_5 \cap \mathcal{C}_6, \quad O \notin \mathcal{C}_7, \quad B^+ \times B^+ \in \mathcal{C}_7 \cap \mathcal{C}_6, \quad B^+ \times B^+ \notin \mathcal{C}_5.$$

Theorem I1 implies that $\mathcal{C}_6$ has the following interesting characterization:

Theorem 4. *$\mathcal{C}_6$ is isomorphic to the lattice of closed equivalence relations on $\text{Ult}(B)$.*

Proof. We first show that $C \models \text{C}_6$ if and only if R_C is transitive. The “only if” part was shown in I14 , so suppose that $C \models \text{C}_6$. Let $\langle F, G \rangle, \langle G, H \rangle \in R_C$, and assume that $\langle F, H \rangle \notin R_C$. Then, $F \times H \not\subseteq C$, and thus, there are $x, y \in B^+$ such that $x \in F, y \in H$, and $x(-C)y$. By C_6 there is some $t \in B$ such that $x(-C)t$ and $t^*(-C)y$. Since $\langle F, G \rangle \in R_C$,

we cannot have $t \in G$, and thus, $t^* \in G$. But $y \in H$ and $\langle G, H \rangle \in R_C$ imply that t^*Cy , a contradiction.

By Theorem 11, there is an isotone one–one correspondence between $\mathcal{C}_6$ and the collection of closed equivalence relations on $\text{Ult}(B)$. Thus, all that remains is to show that the latter is a lattice. It is well known that all equivalence relations on a set form a complete lattice under set inclusion, where the meet is just set intersection, and the join of a family of equivalence relations is the transitive closure of its union. Since an arbitrary intersection of closed sets is closed, and each family of closed equivalence relations has an upper bound, namely, the universal relation on $\text{Ult}(B)$, the collection of all closed equivalence relations on $\text{Ult}B$ is also a complete lattice. $\square$

The following property of clans has been investigated in the theory of proximity spaces and their topological representation, see e.g. [11]:

I_5 . Every maximal clan is a cluster.

It is known that C_6 implies I_5 , and it was unclear whether the converse holds as well. In the following example we will exhibit a contact relation on $FC(\omega)$, that satisfies I_5 , but which satisfies neither C_6 nor C_5 .

Example 1. Suppose that $B = FC(\omega)$; for $n \in \omega$, let F_n be the ultrafilter generated by $\{n\}$; furthermore, let U be the ultrafilter of cofinite sets. Now, define C by

$$C = O \cup \bigcup \{F_n \times F_m : n \equiv m \pmod{2}\}. \quad (2)$$

In other words,

$$xCy \iff x = y \text{ or } (\exists n, m)[n \in x, m \in y, n \equiv m \pmod{2}]. \quad (3)$$

Since each cofinite set contains both odd and even numbers, we have xCy for each cofinite set x and each $y \in B^+$; incidentally, this shows that $C \not\models C_5$. There are exactly two maximal clans in C , namely,

1. $I_0 = \bigcup \{F_n : n \equiv 0 \pmod{2}\} \cup U$,
2. $I_1 = \bigcup \{F_n : n \equiv 1 \pmod{2}\} \cup U$.

Let $x \in B$, and $\{x\} \times I_0 \subseteq C$. If x is cofinite, then $x \in I_0$ by 1. above. If x is finite and contains an even number, say, n , then $x \in F_n \subseteq I_0$. If x is finite and contains only odd numbers, then $x \notin F_n$ for any even n , and also, $x \notin U$. Therefore, $\{x\} \times I_0 \not\subseteq C$. Thus, I_0 is a cluster, and similarly, I_1 is a cluster.

Next, let $x = \{n\}$, where n is even, and set $y = \{n+1\}$; then, $x(-C)y$. Suppose that $z \in B^+$ such that $x(-C)z$; then, in particular, z is finite, i.e. z^* is cofinite, and hence, z^*Cy . This shows that $C \notin \mathcal{C}_6$. $\square$

Turning to C_5 , we make the following observation:

Theorem 5. 1. $\mathcal{C}_5$ is an ideal of $\mathcal{C}$.

2. Let $F, G \in \text{Ult}(B)$, $F \neq G$, and $C = O \cup (F \times G) \cup (G \times F)$. Then, $C \in \mathcal{C}_5$ if and only if neither F nor G are principal.

3. B is isomorphic to a finite-cofinite algebra if and only if $\mathcal{C}_5 = \{O\}$.
4. B is atomless if and only if $\mathcal{C}_5$ contains all atoms of $\mathcal{C}$.

Proof. 1. Clearly $\downarrow \mathcal{C}_5 = \mathcal{C}_5$. Let $C, C' \in \mathcal{C}_5$, and assume that $C \cup C' \notin \mathcal{C}_5$. Then, there exists some $x \in B, x \neq 1$, such that $x(C \cup C')y$ for all $y \in B^+$. Since $C \in \mathcal{C}_5$, there is some $y \neq 0$ such that $x(-C)y$; then, $x \cdot y = 0$ and $xC'y$. Since $C' \in \mathcal{C}_5$, by Lemma 2 there is some $0 \leq z \leq y$ such that $x(-C')z$. But then, xCz , implying xCy , a contradiction. Hence, $C \cup C' \in \mathcal{C}_5$.

2. “ $\Rightarrow$ ”: Suppose that $C \in \mathcal{C}_5$, and assume that w.l.o.g. F is generated by the atom x . Then, $x^* \cdot y \neq 0$ for all $y \notin \{0, x\}$ which implies that x^*Cy for all such y . Since $F \neq G$, we cannot have $x \in G$, hence, $x^* \in G$ and $G \times F \subseteq C$ imply that also x^*Cx .

“ $\Leftarrow$ ”: Suppose that F, G are non-principal, and assume that $C \not\models \mathcal{C}_5$. Then, there is some $x \neq 1$ such that, in particular, xCy for all $y \neq 0, y \leq x^*$. Let w.l.o.g. $x \in F$; then, $B^+ \cap \downarrow x^* \subseteq G$, which implies that G is generated by x ; otherwise, there are nonzero disjoint $y, z \leq x^*$, whose sum is x^* , which cannot be, since $y, z \in G$.

3. The “only if” direction was shown in [17]. Conversely, if $\mathcal{C}_5 = \{O\}$, then, whenever F, G are distinct ultrafilters of B , then $O \cup (F \times G) \cup (G \times F) \notin \mathcal{C}_5$. By 1., this implies that one of F, G must be principal. Hence, B has at most one non-principal ultrafilter, and therefore, B is a finite-cofinite algebra.

4. This follows immediately from the fact that B is atomless if and only if it contains no principal ultrafilters. $\square$

$\mathcal{C}_5$ is generally not generated by the atoms of $\mathcal{C}$: Suppose that $|B| = \kappa \geq \omega$ and that B is atomless. Let $x \in B, x \neq 0, 1$; then, $|\{y : y \leq x\}| = \kappa$ or $|\{y : y \leq x^*\}| = \kappa$. Suppose w.l.o.g. the latter; then, $h(x^*)$ contains a proper closed subset M of cardinality 2^κ . Let $R = h(x) \times M \cup M \times h(x) \cup 1'$; then, R is a closed graph on $\text{Ult}(B)$, and $C_R \models \mathcal{C}_5$.

Finally, turning to $\mathcal{C}_7$, we first note that $\mathcal{C}_7 = \uparrow \mathcal{C}_7$; however, $\mathcal{C}_7$ is, in general, not a filter. To see this, consider the BA with atoms a, b, c , and let F_x be the ultrafilter generated by $x \in \{a, b, c\}$. Then, for $\{x, y\} \subseteq \{a, b, c\}, x \neq y$, the contact relations $O \cup (F_x \times F_y) \cup (F_y \times F_x)$ satisfy $\mathcal{C}_7$, but their meet does not.

However, the situation is brighter when we consider descending chains in $\mathcal{C}_7$:

Lemma 4. *If $\{C_\alpha : \alpha \in I\}$ is a descending chain in $\mathcal{C}_7$, then $\bigcap \{C_\alpha : \alpha \in I\} \in \mathcal{C}_7$.*

Proof. By Theorem 1 it suffices to show that $\bigcap \{C_\alpha : \alpha \in I\} \models \mathcal{C}_7$. If $\langle x, cx \rangle \notin \bigcap \{C_\alpha : \alpha \in I\}$, then $x(-C_\alpha)x^*$ for some $\alpha \in I$. This contradicts $C_\alpha \in \mathcal{C}_7$. $\square$

Thus, by Zorn’s Lemma,

Corollary 3. *For each $C \in \mathcal{C}_7$ there is a minimal $C' \in \mathcal{C}_7$ such that $C' \subseteq C$.*

It was shown in [14] that $C \in \mathcal{C}_7$ if $\langle \text{Ult}(B), R_C \rangle$ is a connected graph, and that the converse is not generally true. It is instructive to recall the example given in [14]:

Example 2. Let $B = FC(\omega)$, and define R on $\text{Ult}(B)$ by

$$R = 1' \cup \{\langle F_n, F_m \rangle : |n - m| = 2\} = 1' \cup \{\langle F_n, F_{n+2} \rangle : n \in \omega\} \cup \{\langle F_{n+2}, F_n \rangle : n \in \omega\}.$$

Clearly, if $|n - m| \neq 2$, then $\langle F_n, F_m \rangle \notin \text{cl}(R)$. Let $x = \{n\}$, and $y = \omega \setminus \{n + 2, n - 2\}$. Then, $x \in F_n, y \in F_\omega$, and thus, $\{F_n\} \times h(y)$ is an open neighbourhood of $\langle F_n, F_\omega \rangle$. Since

$\{n+2, n-2\} \cap y = \emptyset$, $\{F_n\} \times h(y) \times R = \emptyset$, and it follows that $\langle F_n, F_\omega \rangle \notin \text{cl}(R)$; similarly, $\langle F_\omega, F_n \rangle \notin \text{cl}(R)$; hence, R is closed. Let $x \in B, x \neq \emptyset, \omega$. If x is finite, let $m = \max(x)$. Then, $m \in x$ and $m+2 \in x^*$, and therefore, $\langle x, x^* \rangle \in F_m \times F_{m+2}$, i.e. $x C_R x^*$. Hence, C_R is a connected contact relation on B . However, R is not a connected graph, since, for example, there is no path from F_n to F_{n+1} . Indeed, the connected components of R are $\{F_{2n} : n \in \omega\}$ and $\{F_{2n+1} : n \in \omega\}$, each of which is a chain of type ω , and $\{F_\omega\}$. $\square$

If B is finite, the condition is also sufficient:

Theorem 6. *If B is finite, then $C \in \mathcal{C}_7$ implies that R_C is a connected graph.*

Proof. Suppose that M is a connected component of R_C and $M \subsetneq \text{Ult}(B)$. Then, there is no path between any $F_s \in M$ and any $F_t \in \text{Ult}(B) \setminus M$. Let $x = \sum\{s \in \text{At}(B) : F_s \in M\}$ and $y = \sum\{t \in \text{At}(B) : F_t \notin M\}$; then, $x^* = y$. If $x C y$, there are $s, t \in \text{At}(B)$ such that $s \leq x, t \leq y$ and $s C t$, i.e. $\langle F_s, F_t \rangle \in R_C$. This contradicts the fact that F_s and F_t are in different components. $\square$

Since the minimally connected graphs are trees (and vice versa), we obtain

Corollary 4. *If B is finite, then $C \in \mathcal{C}_7$ is minimal if and only if R_C is a tree and $\text{dom}(R_C \setminus 1') = \text{Ult}(B)$.*

Furthermore, since the only connected equivalence relation on $\text{Ult}(B)$ is the universal relation, we have

Lemma 5. *If B is finite, then $\mathcal{C}_6 \cap \mathcal{C}_7 = B^+ \times B^+$.*

Acknowledgement

We would like to thank the referees for careful reading and constructive comments.

References

1. Naimpally, S.A., Warrack, B.D.: Proximity Spaces. Cambridge University Press, Cambridge (1970)
2. de Laguna, T.: Point, line and surface as sets of solids. The Journal of Philosophy 19, 449–461 (1922)
3. Nicod, J.: Geometry in a sensible world. In: Doctoral thesis, Sorbonne, Paris (1924), English translation in Geometry and Induction, Routledge and Kegan Paul (1969)
4. Tarski, A.: Foundation of the geometry of solids. In: Woodger, J.H. (ed.) Logic, Semantics, Metamathematics, pp. 24–29. Clarendon Press, Oxford (1956), Translation of the summary of an address given by A. Tarski to the First Polish Mathematical Congress, Lwów (1927)
5. Whitehead, A.N.: Process and reality. MacMillan, New York (1929)
6. Bennett, B., Düntsch, I.: Algebras, axioms, and topology. In: Aiello, M., van Benthem, J., Pratt-Hartmann, I. (eds.) Handbook of Spatial Logics, pp. 99–159. Kluwer, Dordrecht (2007)
7. Cohn, A.G., Bennett, B., Gooday, J., Gotts, N.M.: Representing and reasoning with qualitative spatial relations about regions. In: Stock, O. (ed.) Spatial and Temporal Reasoning, pp. 97–134. Kluwer, Dordrecht (1997)

8. Balbes, R., Dwinger, P.: *Distributive Lattices*. University of Missouri Press, Columbia (1974)
9. Koppelberg, S.: *General Theory of Boolean Algebras*. Handbook on Boolean Algebras, vol. 1. North Holland, Amsterdam (1989)
10. Engelking, R.: *General Topology*. PWN, Warszawa (1977)
11. Dimov, G., Vakarelov, D.: Contact algebras and region-based theory of space: A proximity approach – I. *Fundamenta Informaticae* 74, 209–249 (2006)
12. Düntsch, I., Winter, M.: A representation theorem for Boolean contact algebras. *Theoretical Computer Science (B)* 347, 498–512 (2005)
13. Dimov, G., Vakarelov, D.: Contact algebras and region-based theory of space: A proximity approach –II. *Fundamenta Informaticae* 74, 251–282 (2006)
14. Düntsch, I., Vakarelov, D.: Region-based theory of discrete spaces: A proximity approach. *Annals of Mathematics and Artificial Intelligence* 49, 5–14 (2007)
15. Dimov, G., Vakarelov, D.: Topological representation of precontact algebras. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) *RelMiCS 2005*. LNCS, vol. 3929, pp. 1–16. Springer, Heidelberg (2006)
16. Galton, A.: The mereotopology of discrete space. In: Freksa, C., Mark, D.M. (eds.) *COSIT 1999*. LNCS, vol. 1661, pp. 251–266. Springer, Heidelberg (1999)
17. Düntsch, I., Winter, M.: Construction of Boolean contact algebras. *AI Communications* 13, 235–246 (2004)

A Non-probabilistic Relational Model of Probabilistic Kleene Algebras

Hitoshi Furusawa¹, Norihiro Tsumagari², and Koki Nishizawa³

¹ Faculty of Science, Kagoshima University
furusawa@sci.kagoshima-u.ac.jp

² Graduate School of Science and Engineering, Kagoshima University
u01tsuma@eni.ac.sci.kagoshima-u.ac.jp

³ Graduate School of Information Sciences, Tohoku University
nishizawa@kb.ecei.tohoku.ac.jp

Abstract. This paper studies basic properties of up-closed multirelations, and then shows that the set of finitary total up-closed multirelations over a set forms a probabilistic Kleene algebra. In Kleene algebras, the star operator is very essential. We investigate the reflexive transitive closure of a finitary up-closed multirelation and show that the closure operator plays a rôle of the star operator of a probabilistic Kleene algebra consisting of the set of finitary total up-closed multirelations as in the case of a Kozen’s Kleene algebra consisting of the set of (usual) binary relations.

1 Introduction

A notion of probabilistic Kleene algebras is introduced by McIver and Weber [7] as a variant of Kleene algebras introduced by Kozen [5]. Using probabilistic Kleene algebras, Cohen’s separation theorems [1] are generalised for probabilistic distributed systems and the general separation results are applied to Rabin’s solution [12] to distributed mutual exclusion with bounded waiting in [8]. This result shows that probabilistic Kleene algebras are useful to simplify a model of probabilistic distributed system without numerical calculations which are usually required and makes difficult to analyse systems when we consider probabilistic behavior.

In this paper we show a non-probabilistic and relational model of probabilistic Kleene algebras. The model consists of the set of finitary total up-closed multirelations on a set. Since multirelations do not have any probabilistic feature, probabilistic Kleene algebras may be applicable to non-probabilistic problems.

Up-closed multirelations are studied as a semantic domain of programs. They serve predicate transformer semantics with both of angelic and demonic non-determinism in the same framework [4, 13, 14]. Also up-closed multirelations provide models of game logic introduced by Parikh [11]. Pauly and Parikh have given an overview of this research area in [10]. Operations of the game logic have been studied from an algebraic point of view by Goranko [3] and Venema [15]. They have given complete axiomatisation of iteration-free game logic. When we see

these applications of up-closed multirelations, it does not seem that a study of the (reflexive) transitive closure deeply relating to iteration of an up-closed multirelation is enough. So we study the notion in this paper.

It is known that the set of (usual) binary relations on a set forms a Kozen's Kleene algebra. Having such a relational model, we can have interpretation of while-programs in a Kleene algebra without any difficulty. Moreover, relational models have suggested a direction of extension of Kleene algebras, for instance, to Kleene algebra with tests [6] and Kleene algebra with domains [2]. Our result shows a possibility of similar extensions of probabilistic Kleene algebras.

2 Probabilistic Kleene Algebra

We recall the definition of probabilistic Kleene algebras introduced in [7].

Definition 1. A *probabilistic Kleene algebra* is a tuple $(K, +, \cdot, *, 0, 1)$ satisfying the following conditions:

$$0 + a = a \quad (1)$$

$$a + b = b + a \quad (2)$$

$$a + a = a \quad (3)$$

$$a + (b + c) = (a + b) + c \quad (4)$$

$$a(bc) = (ab)c \quad (5)$$

$$0a = 0 \quad (6)$$

$$a0 = 0 \quad (7)$$

$$1a = a \quad (8)$$

$$a1 = a \quad (9)$$

$$ab + ac \leq a(b + c) \quad (10)$$

$$ac + bc = (a + b)c \quad (11)$$

$$1 + aa^* \leq a^* \quad (12)$$

$$a(b + 1) \leq a \implies ab^* \leq a \quad (13)$$

$$ab \leq b \implies a^*b \leq b \quad (14)$$

where $\cdot$ is omitted and the order $\leq$ is defined by $a \leq b$ iff $a + b = b$. $\square$

Conditions (10) and (13) are typical ones of probabilistic Kleene algebras. Kozen's Kleene algebras [5] require stronger conditions

$$ab + ac = a(b + c) \text{ and } ab \leq a \implies ab^* \leq a$$

instead of (10) and (13). Clearly, Kozen's Kleene algebras are probabilistic Kleene algebras.

Remark 1. Forgetting two conditions (7) and (13) from probabilistic Kleene algebras, we obtain Möller's lazy Kleene algebras [9]. $\square$

3 Up-Closed Multirelation

In this section we recall definitions and basic properties of multirelations and their operations. More precise information on these can be obtained from [4][13][14].

A *multirelation* R over a set A is a subset of the Cartesian product $A \times \wp(A)$ of A and the power set $\wp(A)$ of A . A multirelation is called *up-closed* if $(x, X) \in R$ and $X \subseteq Y$ imply $(x, Y) \in R$ for each $x \in A$, $X, Y \subseteq A$. The null multirelation $\emptyset$ and the universal multirelation $A \times \wp(A)$ are up-closed, and will be denoted by 0 and ∇ , respectively. The set of up-closed multirelations over A will be denoted by $\text{UMRel}(A)$.

For a family $\{R_i \mid i \in I\}$ of up-closed multirelations the union $\bigcup_{i \in I} R_i$ is up-closed since

$$\begin{aligned}
 & (x, X) \in \bigcup_{i \in I} R_i \text{ and } X \subseteq Y \\
 \iff & \exists i \in I. (x, X) \in R_i \text{ and } X \subseteq Y \\
 \implies & \exists i \in I. (x, Y) \in R_i & (R_i \text{ is up-closed}) \\
 \iff & (x, Y) \in \bigcup_{i \in I} R_i .
 \end{aligned}$$

So $\text{UMRel}(A)$ is closed under arbitrary union $\bigcup$. Then it is immediate that a tuple $(\text{UMRel}(A), \bigcup)$ is a sup-semilattice equipped with the least element 0 with respect to the inclusion ordering $\subseteq$.

$R + S$ denotes $R \cup S$ for a pair of up-closed multirelations R and S . Then the following holds.

Proposition 1. *A tuple $(\text{UMRel}(A), +, 0)$ satisfies conditions (1), (2), (3), and (4) in Definition 1.* $\square$

For a pair of multirelations $R, S \subseteq A \times \wp(A)$ the composition $R; S$ is defined by

$$(x, X) \in R; S \text{ iff } \exists Y \subseteq A. ((x, Y) \in R \text{ and } \forall y \in Y. (y, X) \in S) .$$

It is immediate from the definition that one of the zero laws

$$0 = 0; R$$

are satisfied. The other zero law

$$R; 0 = 0$$

need not hold.

Example 1. Consider the universal multirelation ∇ on a singleton set $\{x\}$. Then, since $(x, \emptyset) \in \nabla$, $\nabla; 0 = \nabla \neq 0$. $\square$

Also the composition $;$ preserves the inclusion ordering $\subseteq$, that is,

$$P \subseteq P' \text{ and } R \subseteq R' \implies P; R \subseteq P'; R'$$

since

$$\begin{aligned}
 (x, X) \in P; R & \iff \exists Y \subseteq A. ((x, Y) \in P \text{ and } \forall y \in Y. (y, X) \in R) \\
 & \implies \exists Y \subseteq A. ((x, Y) \in P' \text{ and } \forall y \in Y. (y, X) \in R') \\
 & \iff (x, X) \in P'; R' .
 \end{aligned}$$

If R and S are up-closed, so is the composition $R; S$ since

$$\begin{aligned} & (x, X) \in R; S \text{ and } X \subseteq Z \\ \implies & \exists Y \subseteq A. ((x, Y) \in R \text{ and } \forall y \in Y. (y, Z) \in S) \quad (S \text{ is up-closed}) \\ \iff & (x, Z) \in R; S \quad . \end{aligned}$$

In other words, the set $\text{UMRel}(A)$ is closed under the composition $;$.

Lemma 1. *Up-closed multirelations are associative under the composition $;$.*

Proof. Let P , Q , and R be up-closed multirelations over a set A . We prove $(P; Q); R \subseteq P; (Q; R)$.

$$\begin{aligned} & (x, X) \in (P; Q); R \\ \iff & \exists Y \subseteq A. ((x, Y) \in P; Q \text{ and } \forall y \in Y. (y, X) \in R) \\ \iff & \exists Y \subseteq A. (\exists Z \subseteq A. ((x, Z) \in P \text{ and } \forall z \in Z. (z, Y) \in Q) \text{ and } \\ & \quad \forall y \in Y. (y, X) \in R) \\ \implies & \exists Z \subseteq A. ((x, Z) \in P \text{ and } \\ & \quad \forall z \in Z. \exists Y \subseteq A. ((z, Y) \in Q \text{ and } \forall y \in Y. (y, X) \in R)) \\ \iff & \exists Z \subseteq A. ((x, Z) \in P \text{ and } \forall z \in Z. (z, X) \in Q; R) \\ \iff & (x, X) \in P; (Q; R) \quad . \end{aligned}$$

For $P; (Q; R) \subseteq (P; Q); R$ it is sufficient to show

$$\begin{aligned} & \exists Z \subseteq A. ((x, Z) \in P \text{ and } \\ & \quad \forall z \in Z. \exists Y \subseteq A. ((z, Y) \in Q \text{ and } \forall y \in Y. (y, X) \in R)) \\ \implies & \exists Y \subseteq A. (\exists Z \subseteq A. ((x, Z) \in P \text{ and } \forall z \in Z. (z, Y) \in Q) \text{ and } \\ & \quad \forall y \in Y. (y, X) \in R) \quad . \end{aligned}$$

Suppose that there exists a set Z such that

$$(x, Z) \in P \text{ and } \forall z \in Z. \exists Y \subseteq A. ((z, Y) \in Q \text{ and } \forall y \in Y. (y, X) \in R) \quad .$$

If Z is empty, it is obvious since we can take the empty set as Y . Otherwise, take a set Y_z satisfying

$$(z, Y_z) \in Q \text{ and } \forall y \in Y_z. (y, X) \in R$$

for each $z \in Z$. Then set $Y_0 = \bigcup_{z \in Z} Y_z$. Since Q is up-closed, $(z, Y_0) \in Q$ for each z . Also $(y, X) \in R$ for each $y \in Y_0$ by the definition of Y_0 . Thus Y_0 satisfies

$$\exists Y \subseteq A. ((x, Z) \in P \text{ and } \forall z \in Z. (z, Y_0) \in Q) \text{ and } \forall y \in Y_0. (y, X) \in R \quad . \quad \square$$

We used the fact that Q is up-closed to show $P; (Q; R) \subseteq (P; Q); R$. Multirelations might not be associative under composition.

Example 2. Consider multirelations

$$\begin{aligned} R &= \{(x, \{x, y, z\}), (y, \{x, y, z\}), (z, \{x, y, z\})\} \text{ and} \\ Q &= \{(x, \{y, z\}), (y, \{x, z\}), (z, \{x, y\})\} \end{aligned}$$

on a set $\{x, y, z\}$. Here, R is up-closed but Q is not. Since $R; Q = 0$, $(R; Q); R = 0$. On the other hand, $R; (Q; R) = R$ since $Q; R = R$ and $R; R = R$. Therefore $(R; Q); R \subseteq R; (Q; R)$ but $R; (Q; R) \not\subseteq (R; Q); R$. Replacing Q with an up-closed multirelation Q' defined by

$$Q' = Q + R ,$$

$R; (Q'; R) = (R; Q'); R$ holds since $Q'; R = R = R; Q'$. □

The *identity* $1 \in \text{UMRel}(A)$ is defined by

$$(x, X) \in 1 \text{ iff } x \in X .$$

Lemma 2. *The identity satisfies the unit laws, that is,*

$$1; R = R \text{ and } R; 1 = R$$

for each $R \in \text{UMRel}(A)$.

Proof. First, we prove $1; R \subseteq R$.

$$\begin{aligned} (x, X) \in 1; R &\iff \exists Y \subseteq A. ((x, Y) \in 1 \text{ and } \forall y \in Y. (y, X) \in R) \\ &\iff \exists Y \subseteq A. (x \in Y \text{ and } \forall y \in Y. (y, X) \in R) \\ &\implies (x, X) \in R . \end{aligned}$$

Conversely, if $(x, X) \in R$, then $(x, X) \in 1; R$ since $(x, \{x\}) \in 1$. Next, we prove $R; 1 \subseteq R$.

$$\begin{aligned} (x, X) \in R; 1 &\iff \exists Y \subseteq A. ((x, Y) \in R \text{ and } \forall y \in Y. (y, X) \in 1) \\ &\iff \exists Y \subseteq A. ((x, Y) \in R \text{ and } \forall y \in Y. y \in X) \\ &\iff \exists Y \subseteq A. ((x, Y) \in R \text{ and } Y \subseteq X) \\ &\implies (x, X) \in R \end{aligned}$$

since R is up-closed. Conversely, if $(x, X) \in R$, then $(x, X) \in R; 1$ since, by the definition of 1 , $(y, X) \in 1$ for each $y \in X$. □

Therefore the following property holds.

Proposition 2. *A tuple $(\text{UMRel}(A), ;, 0, 1)$ satisfies conditions (5), (6), (8), and (9) in Definition 1.* □

As Example 1 has shown, condition (7) need not be satisfied. We discuss about this condition in Section 6

Since the composition $;$ preserves the inclusion ordering $\subseteq$, we have

$$\bigcup_{i \in I} R; S_i \subseteq R; \left(\bigcup_{i \in I} S_i \right)$$

for each up-closed multirelation R and a family $\{S_i \mid i \in I\}$. Also

$$\bigcup_{i \in I} R_i; S = \left(\bigcup_{i \in I} R_i \right); S$$

holds for each up-closed multirelation S and a family $\{R_i \mid i \in I\}$ since

$$\begin{aligned}
 (x, X) \in \bigcup_{i \in I} R_i; S &\iff \exists k. ((x, X) \in R_k; S) \\
 &\iff \exists k. (\exists Y \subseteq A. ((x, Y) \in R_k \text{ and } \forall y \in Y. (y, X) \in S)) \\
 &\iff \exists Y \subseteq A. (\exists k. ((x, Y) \in R_k \text{ and } \forall y \in Y. (y, X) \in S)) \\
 &\iff \exists Y \subseteq A. ((x, Y) \in \bigcup_{i \in I} R_i \text{ and } \forall y \in Y. (y, X) \in S) \\
 &\iff (x, X) \in (\bigcup_{i \in I} R_i); S .
 \end{aligned}$$

Proposition 3. *A tuple $(\text{UMRel}(A), +, ;)$ satisfies conditions [\(I0\)](#) and [\(I1\)](#) in Definition [1](#). $\square$*

The half distributivity [\(I0\)](#) is a typical condition of probabilistic Kleene algebras if we compare with Kozen's Kleene algebras [\[5\]](#) which require also the opposite direction. We give an example showing that the opposite of the half distributivity does not always hold in $\text{UMRel}(A)$.

Example 3. Consider the up-closed multirelation

$$R = \{(x, W) \mid z \in W\} \cup \{(y, W) \mid \{x, z\} \subseteq W\} \cup \{(z, W) \mid \{x, z\} \subseteq W\}$$

on a set $\{x, y, z\}$. Clearly, this R is up-closed. Then, $R; (1 + R) \not\subseteq R; 1 + R$; R since $(y, \{z\}) \notin R; 1 + R$; R though $(y, \{z\}) \in R; (1 + R)$. $\square$

4 Reflexive Transitive Closure

For a (usual) binary relation $r \subseteq A \times A$ on a set A the reflexive transitive closure is given by $\bigcup_{n \geq 0} r^n$ where $r^0 = \{(x, x) \mid x \in A\}$ and $r^{n+1} = r^n; r$. In this section we study the reflexive transitive closure of up-closed multirelations.

First, we give an example showing that $\bigcup_{n \geq 0} R^n$ need not be transitive for each $R \in \text{UMRel}(A)$.

Example 4. We consider the up-closed multirelation R that appeared in Example [3](#). In this case

$$\bigcup_{n \geq 0} R^n = R + 1$$

since $R; R = \{(w, W) \mid w \in \{x, y, z\} \text{ and } \{x, z\} \subseteq W\} \subseteq R$. By the distributive law and the unit law it holds that

$$\begin{aligned}
 (\bigcup_{n \geq 0} R^n); (\bigcup_{n \geq 0} R^n) &= (R + 1); (R + 1) \\
 &= R; (R + 1) + (R + 1) .
 \end{aligned}$$

Since $(y, \{z\}) \in R; (R + 1)$ though $(y, \{z\}) \notin R + 1$, $\bigcup_{n \geq 0} R^n$ is not transitive. $\square$

Next, we give a construction of the reflexive transitive closure of an up-closed multirelation. For $R \in \text{UMRel}(A)$, a mapping $\varphi_R: \text{UMRel}(A) \rightarrow \text{UMRel}(A)$ is defined by

$$\varphi_R(\xi) = R; \xi + 1 .$$

Then, the mapping φ_R preserves the inclusion $\subseteq$. Consider $\bigcup_{n \geq 0} \varphi_R^n(0)$ where φ_R^0 is the identity mapping and $\varphi_R^{n+1} = \varphi_R \circ \varphi_R^n$. Then, $1 \subseteq \bigcup_{n \geq 0} \varphi_R^n(0)$ since $\varphi_R(0) = R; 0 + 1$ and $R \subseteq \bigcup_{n \geq 0} \varphi_R^n(0)$ since $\varphi_R^2(0) = R; (R; 0 + 1) + 1 \supseteq R$.

Lemma 3. $\varphi_R^n(0) \subseteq \varphi_R^{n+1}(0)$ for each $n \geq 0$.

Proof. By induction on n . For $n = 0$ it is trivial since $\varphi_R^0(0) = 0$. Assume that $\varphi_R^n(0) \subseteq \varphi_R^{n+1}(0)$. Then, we have

$$\begin{aligned}\varphi_R^{n+1}(0) &= \varphi_R(\varphi_R^n(0)) \\ &\subseteq \varphi_R(\varphi_R^{n+1}(0)) \\ &= \varphi_R^{n+2}(0)\end{aligned}$$

by the assumption and monotonicity of φ_R . $\square$

Since $(\bigcup_{n \geq 0} \varphi_R^n(0)); (\bigcup_{n \geq 0} \varphi_R^n(0)) = \bigcup_{k \geq 0} \varphi_R^k(0); (\bigcup_{n \geq 0} \varphi_R^n(0))$ by the distributive law, the following property

$$\varphi_R^k(0); (\bigcup_{n \geq 0} \varphi_R^n(0)) \subseteq \bigcup_{n \geq 0} \varphi_R^n(0) \text{ for each } k \geq 0$$

is sufficient to show that $\bigcup_{n \geq 0} \varphi_R^n(0)$ is transitive. However, the property does not hold for every up-closed multirelation.

Definition 2. An up-closed multirelation R is called *finitary* if $(x, Y) \in R$ implies that there exists a finite set Z such that $Z \subseteq Y$ and $(x, Z) \in R$. $\square$

Clearly any multirelations over a finite set are finitary. The set of finitary up-closed multirelations over a set A will be denoted by $\text{UMRel}_f(A)$.

Remark 2. An up-closed multirelation R is called *disjunctive* [10] or *angelic* [4] if, for each $x \in A$ and each $V \subseteq \wp(A)$,

$$(x, \bigcup V) \in R \quad \text{iff} \quad \exists Y \in V. (x, Y) \in R .$$

Let R be disjunctive and $(x, X) \in R$. And let V be the set of finite subsets of X . Then $\bigcup V = X$. By disjunctivity, there exists $Y \in V$ such that $(x, Y) \in R$. Also Y is finite by the definition of V . Therefore disjunctive up-closed multirelations are finitary. However, finitary up-closed multirelations need not be disjunctive. Consider a finitary up-closed multirelation $R = \{(x, \{x, y\})\}$ on a set $\{x, y\}$. Then $\bigcup \{\{x\}, \{y\}\} = \{x, y\}$ and $(x, \{x, y\}) \in R$ but $(x, \{x\}), (x, \{y\}) \notin R$. $\square$

It is obvious that $0, 1 \in \text{UMRel}_f(A)$. Also the set $\text{UMRel}_f(A)$ is closed under arbitrary union $\bigcup$.

Proposition 4. The set $\text{UMRel}_f(A)$ is closed under the composition ;.

Proof. Let P and R be finitary up-closed multirelations. Suppose $(x, X) \in P; R$. Then, by the definition of the composition, there exists $Y \subseteq A$ such that

$$(x, Y) \in P \text{ and } \forall y \in Y. (y, X) \in R .$$

Since P is finitary, there exists a finite set $Y_0 \subseteq Y$ such that

$$(x, Y_0) \in P \text{ and } \forall y \in Y_0. (y, X) \in R .$$

Also, since R is finitary, there exists a finite set $X_y \subseteq X$ such that $(y, X_y) \in R$ for each $y \in Y_0$. Then the set $\bigcup_{y \in Y_0} X_y$ is a finite subset of X such that

$$(x, \bigcup_{y \in Y_0} X_y) \in P; R$$

since $(y, \bigcup_{y \in Y_0} X_y) \in R$ for each $y \in Y_0$. Therefore $P; R$ is finitary. $\square$

Thus, if R is finitary, then so are $\varphi_R^n(0)$ and $\bigcup_{n \geq 0} \varphi_R^n(0)$.

Lemma 4. $\varphi_R^k(0); (\bigcup_{n \geq 0} \varphi_R^n(0)) \subseteq \bigcup_{n \geq 0} \varphi_R^n(0)$ for each $k \geq 0$ if R is finitary.

Proof. By induction on k . For $k = 0$ it is trivial since $\varphi_R^0(0) = 0$ and left-zero law holds. Assume that $\varphi_R^k(0); (\bigcup_{n \geq 0} \varphi_R^n(0)) \subseteq \bigcup_{n \geq 0} \varphi_R^n(0)$. Using the distributive law, the unit law, and this assumption, we have

$$\begin{aligned} \varphi_R^{k+1}(0); (\bigcup_{n \geq 0} \varphi_R^n(0)) &= \varphi_R(\varphi_R^k(0)); (\bigcup_{n \geq 0} \varphi_R^n(0)) \\ &= (R; \varphi_R^k(0) + 1); (\bigcup_{n \geq 0} \varphi_R^n(0)) \\ &= R; \varphi_R^k(0); (\bigcup_{n \geq 0} \varphi_R^n(0)) + \bigcup_{n \geq 0} \varphi_R^n(0) \\ &\subseteq R; (\bigcup_{n \geq 0} \varphi_R^n(0)) + \bigcup_{n \geq 0} \varphi_R^n(0) . \end{aligned}$$

To complete this proof we show $R; (\bigcup_{n \geq 0} \varphi_R^n(0)) \subseteq \bigcup_{n \geq 0} \varphi_R^n(0)$. Suppose $(x, Z) \in R; (\bigcup_{n \geq 0} \varphi_R^n(0))$. Then, since R is finitary, there exists a finite set Y such that

$$(x, Y) \in R \text{ and } \forall y \in Y. \exists k. (y, Z) \in \varphi_R^k(0) .$$

If Y is empty, it is obvious that $(x, Z) \in R$ and we have $(x, Z) \in \bigcup_{n \geq 0} \varphi_R^n(0)$. Otherwise, for each y we take a natural number k_y such that $(y, Z) \in \varphi_R^{k_y}(0)$, and set $k_0 = \sup\{k_y \mid y \in Y\}$. Then, since $\varphi_R^i(0) \subseteq \varphi_R^j(0)$ if $i \leq j$ by Lemma 3, k_0 satisfies

$$\forall y \in Y. (y, Z) \in \varphi_R^{k_0}(0) .$$

Thus, $(x, Z) \in R; \varphi_R^{k_0}(0)$. Also it holds that

$$\begin{aligned} R; \varphi_R^{k_0}(0) &\subseteq R; \varphi_R^{k_0}(0) + 1 \\ &= \varphi_R^{k_0+1}(0) \\ &\subseteq \bigcup_{n \geq 0} \varphi_R^n(0) . \end{aligned}$$

Therefore $(x, Z) \in \bigcup_{n \geq 0} \varphi_R^n(0)$. $\square$

We have already shown that $\bigcup_{n \geq 0} \varphi_R^n(0)$ includes R and is reflexive and transitive if R is finitary. The following property is sufficient to show that $\bigcup_{n \geq 0} \varphi_R^n(0)$ is the least one in the set of reflexive transitive up-closed multirelations including finitary up-closed multirelation R .

Lemma 5. Let R be finitary and $\chi \in \text{UMRel}(A)$ be reflexive, transitive, and including R . Then $\varphi_R^n(0) \subseteq \chi$ for each $n \geq 0$.

Proof. By induction on n . For $n = 0$ it is trivial since $\varphi_R^0(0) = 0$. Assume that $\varphi_R^n(0) \subseteq \chi$. Then we have

$$\begin{aligned} \varphi_R^{n+1}(0) &= R; \varphi_R^n(0) + 1 \\ &\subseteq R; \chi + 1 && (\text{assumption}) \\ &\subseteq \chi; \chi + 1 && (R \subseteq \chi) \\ &\subseteq \chi + 1 && (\chi \text{ is transitive}) \\ &\subseteq \chi && (\chi \text{ is reflexive}) . \end{aligned} \quad \square$$

We have already proved the following.

Theorem 1. $\bigcup_{n \geq 0} \varphi_R^n(0)$ is the reflexive transitive closure of a finitary up-closed multirelation R . $\square$

Remark 3. Though the transitive closure of a (usual) binary relation $r \subseteq A \times A$ is given by $\bigcup_{n \geq 1} r^n$, $\bigcup_{n \geq 1} R^n$ is not always the transitive closure of $R \in \text{UMRel}(A)$. Consider an up-closed multirelation

$$P = \{(x, W) \mid z \in W\} \cup \{(y, W) \mid \{x, z\} \subseteq W\} \cup \{(z, W) \mid \{x, y\} \subseteq W\}$$

on a set $\{x, y, z\}$. Then

$$\bigcup_{n \geq 1} P^n = P + P^2 \text{ and } (\bigcup_{n \geq 1} P^n); (\bigcup_{n \geq 1} P^n) = P(P + P^2) + P^2(P + P^2) .$$

Since $(y, \{x, y\}) \in P; (P + P^2)$ though $(y, \{x, y\}) \notin P + P^2$, $\bigcup_{n \geq 1} P^n$ is not transitive. Next, we give a construction of the transitive closure of a finitary up-closed multirelation. Define a mapping $\psi_R: \text{UMRel}(A) \rightarrow \text{UMRel}(A)$ for $R \in \text{UMRel}(A)$ by

$$\psi_R(\xi) = R; \xi + R .$$

Then it is shown that $\bigcup_{n \geq 0} \psi_R^n(0)$ is the transitive closure of $R \in \text{UMRel}_f(A)$ similarly to the case of the reflexive transitive closure. $\square$

5 The Star

For a finitary up-closed multirelation R we define R^* as

$$R^* = \bigcup_{n \geq 0} \varphi_R^n(0) .$$

In the proof of Lemma 4, $R; R^* \subseteq R^*$ has already been proved. So we have $1 + R; R^* \subseteq R^*$.

Proposition 5. A tuple $(\text{UMRel}_f(A), +, ;, *, 0, 1)$ satisfies condition (12) in Definition 7. $\square$

Two conditions related to the operator $*$ are left to check, namely

$$P; (R+1) \subseteq P \implies P; R^* \subseteq P \quad (15)$$

$$P; R \subseteq R \implies P^*; R \subseteq R \quad (16)$$

for all $P, R \in \text{UMRel}_f(A)$. We show the following properties to show the first implication (15).

Lemma 6. *Let $P, R \in \text{UMRel}_f(A)$. If $P; (R+1) \subseteq P$, then*

1. $\bigcup_{n \geq 0} P; (R+1)^n \subseteq P$, and
2. $\varphi_R^n(0) \subseteq (R+1)^n$ for each $n \geq 0$.

Proof. For 1, it is sufficient to show $P; (R+1)^n \subseteq P$. This is proved by induction on n . For $n = 0$ it is trivial. Assume that $P; (R+1)^n \subseteq P$. Then

$$P; (R+1)^{n+1} \subseteq P; (R+1)^n; (R+1) \subseteq P; (R+1) \subseteq P.$$

2 is also proved by induction on n . For $n = 0$ it is trivial. Assume that $\varphi_R^n(0) \subseteq (R+1)^n$. Then we have

$$\begin{aligned} \varphi_R^{n+1}(0) &= R; \varphi_R^n(0) + 1 \\ &\subseteq R; (R+1)^n + 1 \\ &\subseteq R; (R+1)^n + 1; (R+1)^n \\ &= (R+1); (R+1)^n \\ &= (R+1)^{n+1}. \end{aligned} \quad \square$$

By 1 of Lemma 6 the following property is sufficient to show the first implication (15).

Lemma 7. *For $P, R \in \text{UMRel}_f(A)$ $P; R^* \subseteq \bigcup_{n \geq 0} P; (R+1)^n$ if $P; (R+1) \subseteq P$.*

Proof. Suppose $(x, X) \in P; R^*$. Then, since P is finitary, there exists finite set Y such that

$$(x, Y) \in P \text{ and } \forall y \in Y. \exists k. (y, X) \in \varphi_R^k(0).$$

If Y is empty, it is obvious that $(x, X) \in P$. Otherwise, for each y we take a natural number k_y such that $(y, X) \in \varphi_R^{k_y}(0)$, and set $k_0 = \sup\{k_y \mid y \in Y\}$. Then, since $\varphi_R^i(0) \subseteq \varphi_R^j(0)$ if $i \leq j$ by Lemma 3, k_0 satisfies

$$\forall y \in Y. (y, X) \in \varphi_R^{k_0}(0).$$

Thus $(x, X) \in P; \varphi_R^{k_0}(0)$. Also, by 2 of Lemma 6 $\varphi_R^{k_0}(0) \subseteq (R+1)^{k_0}$. Then we have $P; \varphi_R^{k_0}(0) \subseteq P; (R+1)^{k_0}$. Moreover $P; (R+1)^{k_0} \subseteq \bigcup_{n \geq 0} P; (R+1)^n$. Therefore $(x, X) \in \bigcup_{n \geq 0} P; (R+1)^n$. $\square$

Next, we consider the second implication (16). By the distributivity

$$P^*; R = \left(\bigcup_{n \geq 0} \varphi_P^n(0) \right); R = \bigcup_{n \geq 0} \varphi_P^n(0); R$$

holds. So, for (16) it is sufficient to prove the following property.

Lemma 8. *Let $P, R \in \text{UMRel}_f(A)$. If $P; R \subseteq R$, then $\varphi_P^n(0); R \subseteq R$ for each $n \geq 0$.*

Proof. By induction on n . For $n = 0$ it is trivial since $\varphi_P^0(0) = 0$. Assume that $\varphi_P^n(0); R \subseteq R$. Then we have

$$\begin{aligned} \varphi_P^{n+1}(0); R &= (P; \varphi_P^n(0) + 1); R \\ &= P; \varphi_P^n(0); R + 1; R \\ &\subseteq P; R + R \\ &\subseteq R + R \\ &= R . \end{aligned}$$

□

Proposition 6. *A tuple $(\text{UMRel}_f(A), +, ;, *, 0, 1)$ satisfies conditions (I3) and (I4) in Definition 1.* □

Condition (I3) is typical one of probabilistic Kleene algebras if we compare with Kozen's Kleene algebras [5] which require stronger condition

$$ab \leq a \implies ab^* \leq a$$

instead of (I3). The following example shows that the condition need not hold for finitary up-closed multirelations.

Example 5. Again, consider the up-closed multirelation R appeared in Example 3. $R; R \subseteq R$ is shown in Example 4. Also we have already seen that $(y, \{z\}) \in R; (R + 1)$ in Example 3. Since

$$R; (R + 1) = R; \varphi_R^2(0) \subseteq R; \left(\bigcup_{n \geq 0} \varphi_R^n(0) \right) = R; R^* ,$$

$(y, \{z\}) \in R; R^*$. However, $(y, \{z\}) \notin R$. So, $R; R^* \not\subseteq R$ in spite of $R; R \subseteq R$. □

The following theorem summarises the discussion so far.

Theorem 2. *A tuple $(\text{UMRel}_f(A), +, ;, *, 0, 1)$ satisfies all conditions of probabilistic Kleene algebras except for the right zero law (7).* □

6 The Right Zero Law

In [14] it has been shown that the following notion ensures the right zero law.

Definition 3. A multirelation R on a set A is called *total* if $(x, \emptyset) \notin R$ for each $x \in A$. □

Clearly, the null multirelation 0 and the identity 1 are total.

The set of finitary total up-closed multirelations will be denoted by $\text{UMRel}_f^+(A)$. Then the set $\text{UMRel}_f^+(A)$ is closed under arbitrary union $\bigcup$ and the composition; Since the operator $*$ is defined as a combination of arbitrary union and the composition, $\text{UMRel}_f^+(A)$ is closed under $*$.

Theorem 3. *A tuple $(\text{UMRel}_f^+(A), +, ;, *, 0, 1)$ is not a Kleene algebra in the sense of Kozen [5] but a probabilistic Kleene algebra.* □

The negative result on Kozen's Kleene algebra is induced from either Example 3 or 5 in which we consider only finitary total up-closed multirelations.

7 Conclusion

This paper has studied up-closed multirelations carefully. Then we have shown that the set of finitary total up-closed multirelations is a probabilistic Kleene algebra, where

- the zero element is given by null multirelation,
- the unit element is given by the identity multirelation,
- the addition is given by binary union,
- the multiplication is given by the composition of multirelations, and
- the star is given by the reflexive transitive closure.

The totality has been introduced only for the right zero law. Finitary up-closed multirelations satisfy all conditions of probabilistic Kleene algebras except for the right zero law without assuming the totality.

In addition to this result, comparing with the case of (usual) binary relations, we have investigated the (reflexive) transitive closure of a finitary up-closed multirelation and given its construction. The construction of the reflexive transitive closure provides the star operator.

Acknowledgements

The authors wish to thank Bernhard Möller and Georg Struth for useful comments on an earlier version of this work. The anonymous referees also provided a number of helpful suggestions.

References

1. Cohen, E.: Separation and Reduction. In: Backhouse, R., Oliveira, J.N. (eds.) MPC 2000. LNCS, vol. 1837, Springer, Heidelberg (2000)
2. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. *ACM Trans. Comput. Log.* 7(4), 798–833 (2006)
3. Goranko, V.: The Basic Algebra of Game Equivalences. *Studia Logica* 75(2), 221–238 (2003)
4. Martin, C., Curtis, S., Rewitzky, I.: Modelling Nondeterminism. In: Kozen, D. (ed.) MPC 2004. LNCS, vol. 3125, pp. 228–251. Springer, Heidelberg (2004)
5. Kozen, D.: A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Information and Computation* 110, 366–390 (1994)
6. Kozen, D.: Kleene Algebra with Tests. *ACM Trans. Program. Lang. Syst.* 19(3), 427–443 (1997)
7. McIver, A., Weber, T.: Towards Automated Proof Support for Probabilistic Distributed Systems. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 534–548. Springer, Heidelberg (2005)

8. McIver, A., Cohen, E., Morgan, C.: Using Probabilistic Kleene Algebra for Protocol Verification. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 296–310. Springer, Heidelberg (2006)
9. Möller, B.: Lazy Kleene Algebra. In: Kozen, D. (ed.) MPC 2004. LNCS, vol. 3125, pp. 252–273. Springer, Heidelberg (2004)
10. Pauly, M., Parikh, R.: Game Logic - An Overview. *Studia Logica* 75(2), 165–182 (2003)
11. Parikh, R.: The Logic of Games. *Annals of Discrete Mathematics* 24, 111–140 (1985)
12. Rabin, M.: N-Process Mutual Exclusion with Bounded Waiting by $4 \log_2 N$ -Valued Shared Variable. *JCSS* 25(1), 66–75 (1982)
13. Rewitzky, I.: Binary Multirelations. In: de Swart, H., Orłowska, E., Schmidt, G., Roubens, M. (eds.) *Theory and Applications of Relational Structures as Knowledge Instruments*. LNCS, vol. 2929, pp. 256–271. Springer, Heidelberg (2003)
14. Rewitzky, I., Brink, C.: Monotone Predicate Transformers as Up-Closed Multirelations. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 311–327. Springer, Heidelberg (2006)
15. Venema, Y.: Representation of Game Algebras. *Studia Logica* 75(2), 239–256 (2003)

Increasing Bisemigroups and Algebraic Routing

Timothy G. Griffin and Alexander J.T. Gurney

Computer Laboratory, University of Cambridge
{Timothy.Griffin,Alexander.Gurney}@cl.cam.ac.uk

Abstract. The Internet protocol used today for global routing — the Border Gateway Protocol (BGP) — evolved in a rather organic manner without a clear theoretical foundation. This has stimulated a great deal of recent theoretical work in the networking community aimed at modeling BGP-like routing protocols. This paper attempts to make this work more accessible to a wider community by reformulating it in a purely algebraic setting. This leads to structures we call *increasing bisemigroups*, which are essentially non-distributive semirings with an additional order constraint. Solutions to path problems in graphs annotated over increasing bisemigroups represent locally optimal Nash-like equilibrium points rather than globally optimal paths as is the case with semiring routing.

1 Introduction

A software system can evolve organically while becoming an essential part of our infrastructure. This may even result in a system that is not well understood. Such is the case with the routing protocol that maintains global connectivity in the Internet — the Border Gateway Protocol (BGP). Although it may seem that routing is a well understood problem, we would argue that meeting the constraints of routing between autonomous systems in the Internet has actually given birth to a new class of routing protocols. This class can be characterized by the goal of finding paths that represent locally optimal Nash-like equilibrium points rather than paths that are optimal over all possible paths.

This paper is an attempt to present recent theoretical work on BGP in a purely algebraic setting. Section 2 describes BGP and presents an overview of some of the theoretical work modeling this protocol. Section 3 presents the quadrants model as a framework for discussing how this work relates to the literature on semiring routing. We define *increasing bisemigroups*, which are essentially non-distributive semirings with an additional order constraint. Solutions to path problems in graphs annotated over increasing bisemigroups represent locally optimal Nash-like equilibrium points rather than globally optimal paths as is the case with semiring routing. Section 4 reformulates the work described in Section 2 in terms of increasing bisemigroups. In particular, previous work on BGP modeling has involved reasoning about asynchronous protocols. Here we employ a more traditional approach based on simple matrix multiplication. Section 5 outlines several open problems.

2 Theory and Practice of Interdomain Routing

We can think of routing protocols as being comprised of two distinct components,

$$\text{routing protocol} = \text{routing language} + \text{algorithm},$$

where the protocol's *routing language* is used to configure a network and the (often distributed) *algorithm* is for computing routing solutions to network configurations specified using the routing language. A routing language captures (1) how routes are described, (2) how best routes are selected, (3) how (low-level) policy is described, and (4) how policy is applied to routes.

This characterization of routing protocols may seem straightforward to those familiar with the literature on semiring routing [1234], where we can consider a given semiring to be a routing language. However, the Internet Engineering Task Force (IETF) does not define or develop routing protocols to reflect this thinking. The IETF documents that define protocols (RFCs) tend to present all aspects of a routing protocol algorithmically, mostly due to the emphasis on system performance. The task of untangling the routing language from the routing algorithm for the purposes of analysis is often a very difficult challenge.

Perhaps the most difficult Internet routing protocol to untangle is the Border Gateway Protocol (BGP) [567]. This protocol is used to implement routing in the core of the Internet between Internet Service Providers (ISPs) and large organizations. (The vast majority of corporate and campus networks at the “edge” of the Internet are statically routed to their Internet provider and do not need to run BGP.) At the beginning of 2008 there were over 27,000 autonomous networks using BGP to implement routing in the public Internet¹. An autonomous network can represent anywhere from one to thousands of routers each running BGP. Clearly this protocol is an essential part of the Internet's infrastructure.

The rather complex BGP route selection algorithm can be modeled abstractly as implementing a total pre-order $\leq$ so that if a and b are BGP routes and $a < b$, then a is preferred over b . BGP routes can be thought of as records containing multiple fields, and the order as a lexicographic order with respect to the orders associated with each field's domain. The most significant attribute tends to be used to implement economic relationships between networks, while the less significant tend to be used to implement local *traffic engineering* goals.

Network operators configure routing policies using low-level and vendor-specific languages. Abstractly, a policy can be modeled as a function f that transforms a route a to the route $f(a)$. Policy functions are applied when routes are *exported to* and *imported from* neighboring routers. An important thing to understand is that BGP standards have intentionally underspecified the language used for configuring policy functions. The actual policy languages used today have emerged over the last twenty years from a complex interaction between network operators, router vendors, and protocol engineers. This evolution has taken place with little or no theoretical guidance. This has been positive in the sense that global routing

¹ Each network is associated with a unique identifier that can be found in BGP routing tables. See <http://bgp.potaroo.net>.

was not overly constrained, allowing it to co-evolve along with a viable economic model of packet transport [8].

However, the negative side is that BGP can exhibit serious anomalies. Because of the unconstrained nature of policy functions, routing solutions are not guaranteed to exist and this can lead to protocol divergence [9,10]. Another problem is that routing solutions are not guaranteed to be unique. In an interdomain setting routing policies are considered proprietary and not generally shared between competing ISPs. This can lead to situations where BGP falls into a local optimum that violates the intended policies of operators, yet no one set of operators has enough global knowledge to fix the problem [11].

If BGP policy functions could be constrained to always be *monotonic*, $a \leq b \rightarrow f(a) \leq f(b)$, then standard results might be applied to show that best routes are globally optimal routes and the above mentioned anomalies could not occur. However, it appears very unlikely that any fix imposing monotonicity requirements would be adopted by network operators. Sobrinho has shown that a very simple model of interdomain economic relationships can be implemented with monotonic functions [12,13]. He also showed that more realistic models capturing common implementations of fail-over and load balancing [14] are not monotonic. Yet even if the interdomain world could agree on a monotonic model of interdomain economic relationships, combining this in a monotonic lexicographic order with other common traffic engineering metrics may be impossible. Recent work has shown that obtaining monotonicity with lexicographic products is fairly difficult [15].

One reaction to this situation is to simply declare interdomain routing a “broken mess” and move on to something more tractable. Another is to conclude that there is actually something new emerging here, and that we need to better understand this type of routing and how it relates to more standard approaches.

2.1 The Stable Paths Problem (SPP)

The Stable Paths Problem (SPP) [16,17] was proposed as a simple graph-theoretic model of BGP routing, and was applied to the analysis of several real-world routing problems [14,18,19].

Let $G = (V, E, v_0)$ be a graph with origin v_0 . The set $\mathcal{P}(v, v_0)$ denotes all *simple* paths from node v to node v_0 . For each $v \in V$, $\mathcal{P}^v \subseteq \mathcal{P}(v, v_0)$ denotes the set of *permitted paths* from v to the origin. Let $\mathcal{P}$ be the union of all sets $\mathcal{P}^v$.

For each $v \in V$, there is a non-negative, integer-valued *ranking function* λ^v , defined over $\mathcal{P}^v$, which represents how node v ranks its permitted paths. If $P_1, P_2 \in \mathcal{P}^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be *preferred over* P_1 . Let $\Lambda = \{\lambda^v \mid v \in V - \{v_0\}\}$.

An instance of the *Stable Paths Problem*, $S_{\text{spp}} = (G, \mathcal{P}, \Lambda)$, is a graph together with the permitted paths at each node and the ranking functions for each node. In addition, we assume that $\mathcal{P}^0 = \{(v_0)\}$, and for all $v \in V - \{v_0\}$:

- (empty path is permitted) $\epsilon \in \mathcal{P}^v$,
- (empty path is least preferred) $\lambda^v(\epsilon) = 0$, $\lambda^v(P) > 0$ for $P \neq \epsilon$,

- **(strictness)** If $P_1, P_2 \in \mathcal{P}^v$, $P_1 \neq P_2$, and $\lambda^v(P_1) = \lambda^v(P_2)$, then there is a u such that $P_1 = (v\ u)P'_1$ and $P_2 = (v\ u)P'_2$ (paths P_1 and P_2 have the same next-hop),
- **(simplicity)** If path $P \in \mathcal{P}^v$, then P is a simple path (no repeated nodes),

A *path assignment* is a function π that maps each node $u \in V$ to a path $\pi(u) \in \mathcal{P}^u$. (Note, this means that $\pi(v_0) = (v_0)$.) We interpret $\pi(u) = \epsilon$ to mean that u is not assigned a path to the origin.

The SPP work defines an asynchronous protocol for computing solutions to instances of the stable paths problem. This protocol is in the family of distributed Bellman-Ford algorithms. A sufficient condition (that the *dispute digraph* is acyclic, described below), is shown to imply that this protocol terminates with a locally optimal solution.

The dispute digraph is a directed graph where the nodes are paths in the SPP instance. A *dispute arc* (p, q) represents the situation where

1. $p = (u, v)t$ is a feasible path from u to v_0 with next-hop v ,
2. q is a path from v to v_0 ,
3. either $(u, v)q$ is not feasible at u or p is more preferred than $(u, v)q$ at u .
4. path q is more preferred at v than t .

A *transmission arc* $(p, (u, v)p)$ is defined when p is permitted at v , $(u, v) \in E$, and $(u, v)p$ is permitted at u . The dispute digraph is then the union of dispute and transmission arcs.

Another concept used in [16,17] is the *dispute wheel*. Suppose that p_m ends in the initial node of path p_0 and that p is a cycle $p_0p_2 \cdots p_{m-1}p_m$. Suppose that there are paths q_j , each terminating in v_0 , and each sharing its initial node with p_j . Then this configuration represents a dispute wheel if for each j the path p_jq_{j+1} is more preferred than path q_j , where the subscripts are taken mod m . In [16] it is shown that every dispute wheel can be mapped to a cycle in the dispute digraph.

2.2 Sobrinho's Model

Sobrinho approached the problem from a more algebraic point of view and introduced his routing algebras [20,12]. This work extended his earlier algebraic generalizations of shortest-path routing [21]. Sobrinho's routing algebras take the form $\mathcal{A} = (S, \leq, L, \otimes)$, where $\leq$ is a preference order over S , L is a set of *labels*, and the operator $\otimes$ maps $L \times S$ to S . The set S contains a special element $\infty \in S$ such that: $\sigma < \infty$, for all $\sigma \in S \setminus \{\infty\}$ and $l \otimes \infty = \infty$, for all $l \in L$. A routing algebra $\mathcal{A}$ is said to be *increasing* if $\sigma < l \otimes \sigma$ for each $l \in L$ and each $\sigma \in S - \{\infty\}$.

A (finite) graph $G = (V, E)$ is annotated with a function w which maps edges of E into L . If an initial weight σ_0 is associated with node v_0 , then the weight of a path terminating in v_0 , $p = v_jv_{j-1} \cdots v_1v_0$, is defined to be $w(p) \equiv w(v_j, v_{j-1}) \otimes \cdots \otimes \sigma_0$.

Sobrinho defines an asynchronous protocol for computing solutions to such path problems. Again this protocol is in the family of distributed Bellman-Ford algorithms. The algorithm itself forces paths to be simple — no repetitions of nodes along a path is allowed. Sobrinho develops a sufficient condition (that all cycles are *free*, described below), which guarantees that this protocol terminates with a locally optimal solution. He shows that if an algebra is increasing, then this sufficient condition always holds.

A cycle $v_n v_{n-1} \cdots v_1 v_0 = v_n$ is free if for every $\alpha_0, \alpha_1 \cdots \alpha_n = \alpha_0$, with $\alpha_j \in S - \infty$, there is an i , $1 \leq i \leq n$, such that $\alpha_i < w(u_i, u_{i-1}) \otimes \alpha_{i-1}$. Thus a cycle that is not free is closely related to a dispute wheel of the SPP framework.

3 The Quadrants Model

We first review how path problems are solved using semirings [12,3,4]. Let $S = (S, \oplus, \otimes, 0, 1)$ be a semiring with the additive identity 0, which is also a multiplicative annihilator, and with multiplicative identity 1. We will assume that $\oplus$ is commutative and idempotent. The operations $\oplus$ and $\otimes$ can be extended in the usual way to matrices over S . For example, the multiplicative identity matrix is defined as follows.

$$I(i, j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases}$$

Given a finite directed graph $G = (V, E)$ and a function $w : E \rightarrow S$ we can define the *adjacency* matrix A as

$$A(i, j) = \begin{cases} w(i, j) & \text{if } (i, j) \in E, \\ 0 & \text{otherwise} \end{cases}$$

The *weight* of a path $p = i_1, i_2, i_3, \dots, i_k$ is then calculated as

$$w(p) = w(i_1, i_2) \otimes w(i_2, i_3) \otimes \cdots \otimes w(i_{k-1}, i_k),$$

where the empty path is usually give the weight 1. Define $A^{(k)}$ as

$$A^{(k)} \equiv I \oplus A \oplus A^2 \oplus \cdots \oplus A^k.$$

The following facts are well known. Let $P(i, j)$ be the set of all paths in G from i to j . The set of paths made up of exactly k arcs is denoted by $P^k(i, j) \subseteq P(i, j)$. Then

$$A^k(i, j) = \sum_{p \in P^k(i, j)} w(p).$$

Note that the proof of this fact relies on the (left) distribution rule $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$. The set of paths made up of at most k arcs is denoted by $P^{(k)}(i, j) \subseteq P(i, j)$, and

$$A^{(k)}(i, j) = \sum_{p \in P^{(k)}(i, j)} w(p).$$

In particular, if there exists a q such that $A^{(q)} = A^{(q+1)}$, then

$$A^{(q)}(i, j) = \sum_{p \in P(i, j)} w(p)$$

represents a “global optimum” over all possible paths from i to j .

3.1 Can Iteration be Used to Obtain a “Local” Optimum?

The matrix $B = A^{(q)}$ is a fixed point of the equation

$$B = I \oplus (A \otimes B),$$

which suggests the following *iterative* method of computing $A^{(k)}$.

$$\begin{aligned} A^{[0]} &= I \\ A^{[k+1]} &= I \oplus (A \otimes A^{[k]}) \end{aligned}$$

Of course, using distribution we can see that $A^{(k)} = A^{[k]}$.

However, if distribution does not hold in S we may in some cases still be able to use this iterative method to compute a fixed point! Note that in this case matrix multiplication is not associative.

But how could such a fixed point B be interpreted? For $i \neq j$ we can see that

$$B(i, j) = \sum_{s \in N(i)} w(i, s) \otimes B(s, j)$$

where $N(i)$ is the set of all nodes adjacent to i , $N(i) = \{s \mid (i, s) \in E\}$. Such a fixed point may not represent a “global optimum” yet it can be interpreted as a Nash-like equilibrium point in which each node i obtains “locally optimal” values — node i computes its optimal value associated with paths to node j given only the values adopted by its neighbors. This closely models the type of routing solution we expect for BGP-like protocols.

3.2 Relating Routing Models

We have described the *algebraic* method of computing path weights $w(p)$. The literature on routing also includes the *functional* method, where we have a set of transforms $F \subseteq S \rightarrow S$ and each directed arc (i, j) is associated with a function $f_{(i, j)} \in F$. The weight of a path $p = i_1, i_2, i_3, \dots, i_k$ is then calculated as

$$w(p) = f_{(i_1, i_2)}(f_{(i_2, i_3)}(\dots f_{(i_{k-1}, i_k)}(a) \dots)).$$

where a is some value *originated by the node* i_k . BGP is perhaps the best example of a functional approach to path weight computation.

The literature also contains two methods for path weight summarization. We outlined the *algebraic* approach above using a commutative and idempotent semi-group. The *ordered* method uses an order $\leq$ on S , and we take ‘best weights’ to

mean minimal with respect to $\leq$. These two approaches are closely related (more below), but they are at the same time quite distinct. For example, minimizing the set $S = \{\alpha, \beta\}$ with respect to an order $\leq$ will result in a subset of S , whereas $\alpha \oplus \beta$ may not be an element of S . If α and β are weights associated with network paths p and q , then the best weight $\alpha \oplus \beta$ in the algebraic approach need not be associated with any one network path.

weight computation	weight summarization	
	algebraic	ordered
algebraic	<u>NW — Bisemigroups</u> $(S, \oplus, \otimes)$	<u>NE — Order Semigroups</u> $(S, \leq, \otimes)$
	Semirings [12,3]	Ordered semirings [24,25,26]
	Non-distributive semirings [22,23]	QoS algebras [21]
functional	<u>SW — Semigroup Transforms</u> $(S, \oplus, F)$	<u>SE — Order Transforms</u> $(S, \leq, F)$
	Monoid endomorphisms [12]	Sobrinho structures [12,13].

Fig. 1. The Quadrants Model of Algebraic Routing.

Figure 1 presents the four ways we can combine the algebraic and ordered approaches to weight summarization with the algebraic and functional approaches to weight computation. We discuss each in more detail.

The northwest (NW) quadrant contains bisemigroups of the form $(S, \oplus, \otimes)$. Semirings [12,3] are included in this class, although we do not insist that bisemigroups satisfy the axioms of a semiring. For example, we do not require that $\otimes$ distributes over $\oplus$.

A semigroup $(S, \otimes)$ can be translated to a set of functions using Cayley's left- or right-representation.

$$(S, \otimes) \xrightarrow{\text{cayley}} (S, F)$$

For example, with the left representation we associate a function f_a with each element $a \in S$ and define $f_a(b) = a \otimes b$. The semigroup $(S, \otimes)$ then becomes the set of functions structure $F = \{f_a \mid a \in S\}$. We can then use a Cayley representation to translate a bisemigroup $(S, \oplus, \otimes)$ into a semigroup transform $(S, \oplus, F)$,

$$(S, \oplus, \otimes) \xrightarrow{\text{cayley}} (S, \oplus, F)$$

If we start with a *semiring*, then we arrive in the SW quadrant at what Gondran and Minoux call an *algebra of endomorphisms* [1]. However, it is important to

note that not all semigroup transforms arise in this way from semirings, and we do not require the properties of monoid endomorphisms.

The NE quadrant includes ordered semigroups, which have been studied extensively [24][25][26]. Such structures have the form $(S, \leq, \otimes)$, where $\otimes$ is *monotonic* with respect to $\leq$. That is, if $a \leq b$, then $c \otimes a \leq c \otimes b$ and $a \otimes c \leq b \otimes c$. Sobrinho [21] studied such structures (with total orders) in the context of Internet routing. In our framework, we require only that $\leq$ be a *pre-order* (reflexive and transitive), and we do not require monotonicity but infer it instead (which is why we call these structures *order semigroups* rather than *ordered semigroups*).

Turning to the SE quadrant of Figure 1, we have structures of the form $(S, \leq, F)$, which include Sobrinho's routing algebras [12] as a special case. Sobrinho algebras (as defined in [13]) have the form $(S, \preceq, L, \otimes)$, where $\preceq$ is a *preference relation* over signatures (that is, a total pre-order), L is a set of *labels*, and $\otimes$ is a function mapping $L \times S$ to S . We can map this to an order transform $(S, \preceq, F_L)$ with $F_L = \{g_\lambda \mid \lambda \in L\}$, where $g_\lambda(a) = \lambda \otimes a$. Thus we can think of the pair $(L, \otimes)$ as a means of indexing the set of transforms F_L . In addition to this slightly higher level of abstraction, we do not insist that $\preceq$ be total.

Commutative, idempotent monoids can be translated into orders,

$$(S, \oplus) \xrightarrow{\text{natord}} (S, \leq)$$

in two ways, either $a \leq_R^\oplus b \equiv b = a \oplus b$, or $a \leq_L^\oplus b \equiv a = a \oplus b$. These orders are clearly dual, with $a \leq_L b$ iff $b \leq_R a$. If 1 is also an additive annihilator, then we have for all $a \in S$, $0 \leq_R^\oplus a \leq_R^\oplus 1$ and $1 \leq_L^\oplus a \leq_L^\oplus 0$, and the orders are *bounded*.

Using the *natord* and *cayley* translations we can move from the NW to the SE quadrants of Figure 1,

$$\begin{array}{ccc} (S, \oplus, \otimes) & \xrightarrow{\text{natord}} & (S, \leq, \otimes) \\ \downarrow \text{cayley} & & \downarrow \text{cayley} \\ (S, \oplus, F) & \xrightarrow{\text{natord}} & (S, \leq, F) \end{array}$$

We can use these translations to investigate how properties appropriate to each quadrant are related. For example, an order transform is *increasing* when for all a and f we have

$$a \neq \top \implies a < f(a),$$

where $\top$ is the top element of the order. Pushing this property through the above translations yields a definition of increasing for each quadrant.

$$\begin{array}{ccc} (a \neq 0 \implies a = a \oplus (b \otimes a)) \wedge & \xrightarrow{\text{left-natord}} & a \neq \top \implies a < b \otimes a \\ (b \otimes a = a \oplus (b \otimes a) \implies a = 0) & & \downarrow \text{left-cayley} \\ \downarrow \text{left-cayley} & & \\ (a \neq 0 \implies a = a \oplus f(a)) \wedge & \xrightarrow{\text{left-natord}} & a \neq \top \implies a < f(a) \\ (f(a) = a \oplus f(a) \implies a = 0) & & \end{array}$$

For example, a *left increasing bisemigroup* is a bisemigroup where for all a and b we have $a \neq 0 \implies a = a \oplus (b \otimes a)$ and $b \otimes a = a \oplus (b \otimes a) \implies a = 0$. In other words, where $a \neq 0 \implies a <_L^\oplus b \otimes a$. In this paper we will use the term *increasing bisemigroup* to mean left increasing bisemigroup.

3.3 Quadrants Model and Metarouting

Griffin and Sobrinho [13] proposed *metarouting* as a means of defining routing protocols in a high-level and declarative manner. Metarouting is based on using a *metalanguage* to specify routing languages. Algebraic properties required by algorithms are derived automatically from a metalanguage specification, in much the same way that types are derived in modern programming languages.

It is envisioned that metarouting will be used to specify (and implement) new routing protocols as follows. Assume that a fixed menu of generic routing algorithms has been implemented, each associated with a specific set of correctness requirements. First, the algebraic component is defined using the metalanguage, resulting in a set of automatically inferred properties. Next, the routing language can then be associated with any algorithm whose requirements set is contained in the set of inferred properties. This checking could be done at protocol design time or later at network configuration time. A metarouting implementation must then compile the specification and algorithm choices into efficient code for representing routing tables, calculating best routes, parsing and packing binary on-the-wire representations and so on. Protocol compilation is a topic of ongoing research.

The quadrants model of Figure 1 has been adopted as the algebraic basis for metarouting. Rather than confining metarouting to the SE quadrant, as was done in [13], the metarouting project is now attempting to capture structures and operations in each of the four quadrants, as well as operations between quadrants. In this model, properties are not required but inferred.

4 A Relational Reformulation in Terms of Bisemigroups

We reformulate the theories described in Section 2 in terms of bisemigroups. This is not meant to be completely faithful in every detail, rather it represents an attempt to recast the essential ideas in a purely algebraic setting.

Let $S = (S, \oplus, \otimes)$ be a bisemigroup. Throughout this section we will assume that $\oplus$ is idempotent, commutative, and selective ($a \oplus b = a \vee a \oplus b = b$), and that both 0 and 1 exist and that 0 is a multiplicative annihilator. Note that since $\oplus$ is idempotent, commutative, and selective it follows that $\leq_L^\oplus$ is a total order.

Let A be an adjacency matrix over S . Since $\oplus$ is selective, for each $i \neq j$ there exists $s_{(i,j)}^k \in N(i) \equiv \{s \mid (i, s) \in E\}$ such that

$$A^{[k+1]}(i, j) = \sum_{s \in N(i)} w(i, s) \otimes B(s, j) = w(i, s_{(i,j)}^k) \otimes A^{[k]}(s_{(i,j)}^k, j)$$

We assume that we have a deterministic method of selecting a unique $s_{(i,j)}^k$.

For the iterative algorithm we define a particular sequence of values that is called *the history of* $A^{[k]}(i, j)$. Histories are inspired by constructs of the same name in [27] that record causal chains of events in an asynchronous protocol. Here, the history of $A^{[k]}(i, j)$, denoted $H^{[k]}(i, j)$, will in some sense explain how the value $A^{[k]}(i, j)$ came to be adopted at step k of the iteration.

$$H^{[0]}(i, j) = (1)$$

$$H^{[k+1]}(i, j) = \begin{cases} H^{[k]}(i, j) & \text{if } A^{[k]}(i, j) = A^{[k+1]}(i, j), \\ H^{[k]}(s_{(i,j)}^k, j), A^{[k+1]}(i, j) & \text{if } A^{[k+1]}(i, j) <_L^\oplus A^{[k]}(i, j), \\ H^{[k]}(s_{(i,j)}^{k-1}, j), A^{[k]}(i, j) & \text{if } A^{[k]}(i, j) <_L^\oplus A^{[k+1]}(i, j). \end{cases}$$

Note that if $A^{[k+1]}(i, j) <_L^\oplus A^{[k]}(i, j)$, then node i obtained a more preferred value at step $k + 1$. In this case the history $H^{[k+1]}(i, j)$ is the sequence $H^{[k]}(s_{(i,j)}^k, j), A^{[k+1]}(i, j)$, where $H^{[k]}(s_{(i,j)}^k, j)$ is a history explaining how value $A^{[k]}(s_{(i,j)}^k, j)$ was adopted at state k . Since $A^{[k+1]}(i, j) = w(i, s_{(i,j)}^k) \otimes A^{[k]}(s_{(i,j)}^k, j)$, the complete history explains how $A^{[k+1]}(i, j)$ was adopted at step $k + 1$.

On the other hand, when $A^{[k]}(i, j) <_L^\oplus A^{[k+1]}(i, j)$, then node i lost a more preferred value at step $k + 1$. In this case the history $H^{[k+1]}(i, j)$ is the sequence $H^{[k]}(s_{(i,j)}^{k-1}, j), A^{[k]}(i, j)$, which ends in the value lost at step $k + 1$. Since this lost value is $A^{[k]}(i, j) = w(i, s_{(i,j)}^{k-1}) \otimes A^{[k-1]}(s_{(i,j)}^{k-1}, j)$, the sequence $H^{[k]}(s_{(i,j)}^{k-1}, j)$ explains how node $s_{(i,j)}^{k-1}$ came to adopt $A^{[k]}(s_{(i,j)}^{k-1}, j)$ at step k , thus forcing node i to abandon $A^{[k]}(i, j)$ at step $k + 1$.

Of course this last type of history depends on *violations* of monotonicity,

$$\forall a, b, c \in S : a \leq_L^\oplus b \rightarrow c \otimes a \leq_L^\oplus c \otimes b.$$

We define the *dispute relation* D_S to record such violations,

$$D_S \equiv \{(a, c \otimes b) \mid a, b, c \in S, a \leq_L^\oplus b \wedge c \otimes b <_L^\oplus c \otimes a\}$$

Of course, in the case that S is monotonic, then D_S is empty. In addition we define a relation

$$T_S \equiv \{(a, b \otimes a) \mid a, b \in S, b \neq 1\}.$$

Note that T_S is the anti-reflexive sub-relation of $\leq_R^\otimes$, (using $\otimes!$) where

$$a \leq_R^\otimes b \equiv \exists c \in S : b = c \otimes a.$$

The *generalized dispute digraph* is then defined as the relation

$$\mathfrak{D}_S = (T_S \cup D_S)^{tc},$$

where tc denotes the transitive closure.

Note that if $(a, b \otimes a) \in T_S$, then if S is increasing we have $a <_L^\oplus b \otimes a$. If $(a, c \otimes b) \in D_S$, then $a \leq_L^\oplus b$, and if S is increasing then $b <_L^\oplus c \otimes b$, so $a <_L^\oplus c \otimes b$. Thus we have proved the following.

Lemma 1. *If S is increasing, then $\mathfrak{D}_S \subseteq <_L^\oplus$.*

A $\mathfrak{D}_S$ sequence σ is any non-empty sequence of values over S such that if $\sigma = a_1, a_2, \dots, a_k$, for $2 \leq k$, then for each $1 \leq i < k$ we have $(a_i, a_{i+1}) \in \mathfrak{D}_S$.

Lemma 2. *For each k, i , and j , $H^{[k]}(i, j)$ is a $\mathfrak{D}_S$ sequence.*

Lemma 3. *Suppose that $A^{[k]}(i, j) \neq A^{[k+1]}(i, j)$, then $|H^{[k+1]}(i, j)| = k + 1$.*

Theorem 1. *If S is an increasing bisemigroup and only simple paths are allowed, then there must exist a k such that $A^{[k]} = A^{[k+1]}$. Thus $B = A^{[k]}$ is a solution to the equation $B = I \oplus (A \otimes B)$.*

As mentioned in Section 2, the SPP theory also used the concept of *dispute wheels* while Sobrinho's theory used the related concept of *non-free cycles*. We now show how these concepts are related to generalized dispute digraphs.

Dispute wheels and non-free cycles can both be captured relationally [28]. Let

$$\mathfrak{R}_S \equiv (\leq_R^\otimes \circ <_L^\oplus)^{tc}.$$

Lemma 4. *Suppose that $a_1 \mathfrak{R}_S a_2 \mathfrak{R}_S a_3$. That is, there exists b_1 and b_2 such that*

$$a_1 \leq_R^\otimes b_1 \otimes a_1 <_L^\oplus a_2 \leq_R^\otimes b_2 \otimes a_2 <_L^\oplus a_3.$$

Then either $a_1 \leq_R^\otimes a_3$ or $(b_1 \otimes a_1, b_2 \otimes a_2) \in \mathfrak{D}_S$.

Corollary 1. *If $(a, a) \in \mathfrak{R}_S$, then $(a, a) \in \mathfrak{D}_S$.*

In particular, if S is an increasing bisemigroup, then we know that all cycles are free and that dispute wheels cannot exist.

5 Open Problems and Discussion

We do not mean to suggest that the only possible application of increasing bisemigroups is in network routing. Non-distributive semirings have been considered in other types of path optimization problems such as circuit layout [22,23], and there may be problems in areas such as operations research to which increasing bisemigroups could be applied. This suggests several open problems.

5.1 Problem 1: Dropping Selectivity

To what extent can the results of the previous section be extended to non-selective bisemigroups?

The assumption that $\leq_L^\oplus$ is a total order pervades the proof techniques we use. However, there is good motivation for relaxing the totality condition and allowing for a non-selective $\oplus$. This is important for the metarouting effort [13], since many of the translations going from eastern to western quadrants of Figure 1 involve a *min-set construction*, which does not, in general, result in an additive semigroup that is selective.

Min-set constructions are a type of *reduction* defined by Wongseelashote [29]. For any finite subset $A \subseteq S$, let $\min_{\leq}(A) \equiv \{x \in A \mid \forall y \in A : \neg(y < x)\}$, be the *minimal subset of A*. Here $y < x$ means $y \leq x \wedge \neg(x \leq y)$ and so the operation is well defined even for pre-orders. The set of all minimal sets is denoted as $\min_{\leq}(S) \equiv \{A \subseteq S \mid A \text{ is finite and } \min_{\leq}(A) = A\}$. If $A, B \in \min_{\leq}(S)$, then define $A \oplus B \equiv \min_{\leq}(A \cup B)$. Thus we can construct a commutative and idempotent semigroup $(\min_{\leq}(S), \oplus)$ from a pre-ordered set $(S, \leq)$.

If $a \neq b$ and both are in a minimal set $A = \min_{\leq}(A)$, then either they are equivalent $a \sim b$ ($a \leq b$ and $b \leq a$), or they are incomparable $a \nparallel b$ ($\neg(a \leq b)$ and $\neg(b \leq a)$). We believe that min-set semigroups closely model the way Internet routing protocols compute *equal cost multi-paths* and the way they can partition routes into distinct *service classes*. Equal cost multi-paths arise when the weights of at least two distinct paths are equivalent, $w(p) \sim w(q)$. *Load balancing* can then be implemented by forwarding traffic along both paths p and q (today this is usually accomplished with a function that selects paths by hashing on information such as IP addresses and port numbers). In the case that $w(p) \nparallel w(q)$, then we can interpret this as meaning that the *data traffic itself* must contain information that can be used to select path p or path q . As a simple example, suppose that weights $w(p)$ somehow contain a destination address and that $w(p) \nparallel w(q)$ arises only when these addresses differ. In this case the destination address carried in a data packet is used to select a path. For another example, suppose that weights $w(p)$ contain a *type of service* and that $w(p) \nparallel w(q)$ means the associated paths support different types of service. In this case the data traffic would be expected to contain a type-of-service field used to select an appropriate path.

5.2 Problem 2: Complexity Bounds

What is the computational complexity (number of steps required) of the iterative algorithm for increasing bisemigroups? We suspect that the worst case complexity will involve an exponential in the number of nodes in the graph. However, this may not be the case for all (non-distributive) increasing bisemigroups.

As mentioned, previous complexity analysis of BGP has invariably involved distributed (asynchronous) algorithms. Yet an asynchronous version of our iterative algorithm can have exponential worst-case complexity even in the case of shortest-paths routing due to the non-deterministic interleaving of routing messages (see for example [30]). Here we are asking instead for the inherent complexity associated with an increasing bisemigroups, in terms of the complexity of our iterative algorithm alone.

Acknowledgments

This paper benefited greatly from discussions with Gordon Wilfong and João Luís Sobrinho. We also thank John Billings, Martin Hyland, Philip Taylor, and

Barney Stratford for their helpful comments. A. Gurney is supported by a Doctoral Training Account from the Engineering and Physical Sciences Research Council (EPSRC). T. Griffin is grateful for support under the the Cisco Collaborative Research Initiative.

References

1. Gondran, M., Minoux, M.: Graphes, diïodes et semi-anneaux: Nouveaux modèles et algorithmes. Tec & Doc (2001)
2. Gondran, M., Minoux, M.: Graphs and Algorithms. Wiley, Chichester (1984)
3. Carré, B.: Graphs and Networks. Oxford University Press, Oxford (1979)
4. Backhouse, R., Carr, B.: Regular algebra applied to path-finding problems. *J. Inst. Math. Appl.* 15, 161–181 (1975)
5. Rekhter, Y., Li, T.: A Border Gateway Protocol. RFC 1771 (BGP version 4) (March 1995)
6. Stewart, J.W.: BGP4: Inter-Domain Routing in the Internet. Addison-Wesley, Reading (1999)
7. Halabi, S., McPherson, D.: Internet Routing Architectures, 2nd edn. Cisco Press (2001)
8. Huston, G.: Interconnection, peering and settlements: Parts I and II. *Internet Protocol Journal* 2(1 and 2) (March, June 1999)
9. Varadhan, K., Govindan, R., Estrin, D.: Persistent route oscillations in inter-domain routing. *Computer Networks* 32, 1–16 (2000) (based on a 1996 technical report)
10. Systems, C.: Endless BGP convergence problem in Cisco IOS software releases. Field Note, October 10 (2001), <http://www.cisco.com/warp/public/770/fn12942.html>
11. Griffin, T.G., Huston, G.: RFC 4264: BGP Wedgies, IETF (November 2005)
12. Sobrinho, J.L.: An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking* 13(5), 1160–1173 (2005)
13. Griffin, T.G., Sobrinho, J.L.: Metarouting. In: *Proc. ACM SIGCOMM* (August 2005)
14. Griffin, T.G., Gao, L., Rexford, J.: Inherently safe backup routing with BGP. In: *Proc. IEEE INFOCOM* (April 2001)
15. Gurney, A., Griffin, T.G.: Lexicographic products in metarouting. In: *Proc. Inter. Conf. on Network Protocols* (October 2007)
16. Griffin, T.G., Shepherd, F.B., Wilfong, G.: Policy disputes in path-vector protocols. In: *Proc. Inter. Conf. on Network Protocols* (November 1999)
17. Griffin, T.G., Shepherd, F.B., Wilfong, G.: The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking* 10(2), 232–243 (2002)
18. Griffin, T.G., Wilfong, G.: On the correctness of IBGP configuration. In: *Proc. ACM SIGCOMM* (September 2002)
19. Griffin, T.G., Wilfong, G.: An analysis of the MED oscillation problem in BGP. In: *Proc. Inter. Conf. on Network Protocols* (2002)
20. Sobrinho, J.L.: Network routing with path vector protocols: Theory and applications. In: *Proc. ACM SIGCOMM* (September 2003)
21. Sobrinho, J.L.: Algebra and algorithms for QoS path computation and hop-by-hop. *IEEE/ACM Transactions on Networking* 10(4), 541–550 (2002)

22. Lengauer, T., Theune, D.: Unstructured path problems and the making of semirings. In: Dehne, F., Sack, J.-R., Santoro, N. (eds.) WADS 1991. LNCS, vol. 519, pp. 189–200. Springer, Heidelberg (1991)
23. Lengauer, T., Theune, D.: Efficient algorithms for path problems with general cost criteria. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 314–326. Springer, Heidelberg (1991)
24. Fuchs, L.: Partially Ordered Algebraic Systems. Addison-Wesley, Reading (1963)
25. Birkhoff, G.: Lattice Theory, 3rd edn. Amer. Math. Soc., Providence, RI (1967)
26. Johnson, R.E.: Free products of ordered semigroups. Proceedings of the American Mathematical Society 19(3), 697–700 (1968)
27. Griffin, T., Wilfong, G.: A safe path vector protocol. In: Proc. IEEE INFOCOM (March 2000)
28. Chau, C., Gibbens, R., Griffin, T.G.: Towards a unified theory of policy-based routing. In: Proc. IEEE INFOCOM (April 2006)
29. Wongseelashote, A.: Semirings and path spaces. Discrete Mathematics 26(1), 55–78 (1979)
30. Karloff, H.: On the convergence time of a path-vector protocol. In: ACM-SIAM Symposium on Discrete Algorithms (SODA) (2004)

A Proofs

Lemma 3. The proof is by induction on k . The base case is clear. Suppose every entry of $H^{[k]}$ is a $\mathfrak{D}_S$ sequence. The analysis of $H^{[k+1]}(i, j)$ is in three cases.

Case 1 : $A^{[k]}(i, j) = A^{[k+1]}(i, j)$. Then $H^{[k+1]}(i, j) = H^{[k]}(i, j)$ and the claim holds.

Case 2 : $A^{[k+1]}(i, j) <_L^\oplus A^{[k]}(i, j)$, so we have

$$\begin{aligned} w(i, s_{(i,j)}^k) \otimes A^{[k]}(s_{(i,j)}^k, j) &<_L^\oplus w(i, s_{(i,j)}^{k-1}) \otimes A^{[k-1]}(s_{(i,j)}^{k-1}, j) \\ &\leq_L^\oplus w(i, s_{(i,j)}^k) \otimes A^{[k-1]}(s_{(i,j)}^k, j). \end{aligned}$$

In this case $H^{[k+1]}(i, j) = H^{[k]}(s_{(i,j)}^k, j)$, $A^{[k+1]}(i, j)$. There are three sub-cases to consider.

Case 2.1 : $A^{[k-1]}(s_{(i,j)}^k, j) = A^{[k]}(s_{(i,j)}^k, j)$. This is not possible.

Case 2.2 : $A^{[k]}(s_{(i,j)}^k, j) <_L^\oplus A^{[k-1]}(s_{(i,j)}^k, j)$. Then $(A^{[k]}(s_{(i,j)}^k, j), w(i, s_{(i,j)}^k) \otimes A^{[k]}(s_{(i,j)}^k, j))$ is in T_S , and since $H^{[k]}(s_{(i,j)}^k, j)$ ends in $A^{[k]}(s_{(i,j)}^k, j)$, it follows that $H^{[k+1]}(i, j)$ is a $\mathfrak{D}_S$ sequence.

Case 2.3 : $A^{[k-1]}(s_{(i,j)}^k, j) <_L^\oplus A^{[k]}(s_{(i,j)}^k, j)$. Then $(A^{[k-1]}(s_{(i,j)}^k, j), A^{[k+1]}(i, j))$ is in D_S , and since $H^{[k]}(s_{(i,j)}^k, j)$ ends in the value $A^{[k-1]}(s_{(i,j)}^k, j)$, it follows that $H^{[k+1]}(i, j)$ is a $\mathfrak{D}_S$ sequence.

Case 3 : $A^{[k]}(i, j) <_L^\oplus A^{[k+1]}(i, j)$, so we have

$$\begin{aligned} w(i, s_{(i,j)}^{k-1}) \otimes A^{[k-1]}(s_{(i,j)}^{k-1}, j) &<_L^\oplus w(i, s_{(i,j)}^k) \otimes A^{[k]}(s_{(i,j)}^k, j) \\ &\leq_L^\oplus w(i, s_{(i,j)}^{k-1}) \otimes A^{[k]}(s_{(i,j)}^{k-1}, j). \end{aligned}$$

In this case $H^{[k+1]}(i, j) = H^{[k]}(s_{(i,j)}^{k-1}, j)$, $A^{[k]}(i, j)$. There are three sub-cases to consider.

Case 3.1 : $A^{[k-1]}(s_{(i,j)}^{k-1}, j) = A^{[k]}(s_{(i,j)}^{k-1}, j)$. This is not possible.

Case 3.2 : $A^{[k]}(s_{(i,j)}^{k-1}, j) <_L^\oplus A^{[k-1]}(s_{(i,j)}^{k-1}, j)$. Then

$$(A^{[k]}(s_{(i,j)}^{k-1}, j), w(i, s_{(i,j)}^{k-1}) \otimes A^{[k-1]}(s_{(i,j)}^{k-1}, j)) \in D_S,$$

and since $H^{[k]}(s_{(i,j)}^{k-1}, j)$ ends in $A^{[k]}(s_{(i,j)}^{k-1}, j)$, $H^{[k+1]}(i, j)$ is a $\mathfrak{D}_S$ sequence.

Case 3.3 : $A^{[k-1]}(s_{(i,j)}^{k-1}, j) <_L^\oplus A^{[k]}(s_{(i,j)}^{k-1}, j)$. Then $H^{[k]}(s_{(i,j)}^{k-1}, j)$ ends in the value $A^{[k-1]}(s_{(i,j)}^{k-1}, j)$, and

$$(A^{[k-1]}(s_{(i,j)}^{k-1}, j), w(i, s_{(i,j)}^{k-1}) \otimes A^{[k-1]}(s_{(i,j)}^{k-1}, j)) \in T_S,$$

so $H^{[k+1]}(i, j)$ is a $\mathfrak{D}_S$ sequence.

Lemma 3. The proof is by induction on k . For $k = 0$, suppose $A^{[0]}(i, j) \neq A^{[1]}(i, j)$. Since $A^{[1]}(i, j) = w(i, s_{(i,j)}^0) \otimes A^{[0]}(s_{(i,j)}^0, j) = w(i, s_{(i,j)}^0) \otimes I(s_{(i,j)}^0, j)$ it must be that $s_{(i,j)}^0 = j$ and $A^{[1]}(i, j) = w(i, j)$. Therefore $H^{[1]}(i, j) = 1$, $w(i, j)$, and $|H^{[1]}(i, j)| = k + 1$.

Next, suppose that $A^{[k]}(i, j) \neq A^{[k+1]}(i, j)$. There are two cases to consider.

Case 1 : $A^{[k+1]}(i, j) <_L^\oplus A^{[k]}(i, j)$. In this case

$$H^{[k+1]}(i, j) = H^{[k]}(s_{(i,j)}^k, j), A^{[k+1]}(i, j).$$

As in the proof of Lemma 3, it must be that $A^{[k-1]}(s_{(i,j)}^k, j) \neq A^{[k]}(s_{(i,j)}^k, j)$.

By induction, $|H^{[k]}(s_{(i,j)}^k, j)| = k$, so $|H^{[k+1]}(i, j)| = k + 1$.

Case 2 : $A^{[k]}(i, j) <_L^\oplus A^{[k+1]}(i, j)$, so we have

$$H^{[k+1]}(i, j) = H^{[k]}(s_{(i,j)}^{k-1}, j), A^{[k]}(i, j).$$

As in the proof of Lemma 3, it must be that $A^{[k-1]}(s_{(i,j)}^{k-1}, j) \neq A^{[k]}(s_{(i,j)}^{k-1}, j)$.

By induction, $|H^{[k]}(s_{(i,j)}^{k-1}, j)| = k$, so $|H^{[k+1]}(i, j)| = k + 1$.

Theorem 1. Suppose that k does not exist. Since only simple paths are allowed, the set of values $w(p)$ for all paths p is finite. Since histories must grow without bound there must at some point be an a such that $(a, a) \in \mathfrak{D}_S$, which contradicts Lemma 1.

Lazy Relations

Walter Guttman

Institut für Programmiermethodik und Compilerbau
Universität Ulm, 89069 Ulm, Germany
walter.guttman@uni-ulm.de

Abstract. We present a relational model of non-strict computations in an imperative, non-deterministic context. Undefinedness is represented independently of non-termination. The relations satisfy algebraic properties known from other approaches to model imperative programs; we introduce additional laws that model dependence in computations in an elegant algebraic form using partial orders. Programs can be executed according to the principle of lazy evaluation, otherwise known from functional programming languages. Local variables are treated by relational parallel composition.

1 Introduction

Our goal is to develop a relational model of non-strict computations in an imperative, non-deterministic context. As a simple motivation of the issues we are about to address, consider the statement $P =_{\text{def}} x_1, x_2 := \frac{1}{0}, 2$ that simultaneously assigns an undefined value to x_1 and 2 to x_2 . In a conventional language its execution aborts, but we want undefined expressions to remain harmless if their value is not needed. This is standard in functional programming languages with lazy evaluation like Haskell [17]. Yet also in an imperative language it can be reasonable to require that $P ; x_1 := x_2 = x_1, x_2 := 2, 2$ holds since the value of x_1 after the execution of P is never used. To see this, consider the following Haskell program that implements $P ; x_1 := x_2$ in monadic style:

```
import Data.IORef;
main = do r <- newIORef (div 1 0 , 2)
        modifyIORef r (\(x1,x2) -> (x2,x2))
        x <- readIORef r
        print x
```

It prints (2,2) terminating successfully, but would abort if (x2,x2) was changed to (x1,x1). Integrating non-determinism additionally is useful for program specification and development.

Let us describe our new approach which has these qualities. As usual, we represent undefinedness of individual variables by adding a special value $\perp$ to their ranges. We add another special element ∞ to distinguish non-termination from undefinedness. The difficulty is to choose the relations and operations (that model computations) such that, on the one hand, they handle these special values

correctly and, on the other hand, they are continuous. The latter is required to iteratively approximate the solutions to recursive equations, which corresponds to the evaluation of recursion in practice. Furthermore, key constructs such as composition and choice should retain their familiar relational meaning to obtain nice algebraic properties. We solve this problem by introducing a partial order on the ranges of variables and states, and forming the closure of relations with respect to this order.

Section 3 presents a compendium of relations modelling a selection of programming constructs. We identify several algebraic properties they satisfy, starting with isotony and the left and right unit laws. In Section 3.2 we derive further properties, namely finite branching, continuity and totality. We thus obtain a theory similar to that of existing approaches, but describing non-strict computations, able to yield defined results in spite of undefined inputs. Moreover, it is sufficient to execute only those parts of a program necessary to calculate the final results, which can improve efficiency.

With lazy execution comes the need to consider dependences between individual computations. Such dependences also play a role in optimising program transformations like those performed in compilers. Their structure is investigated in Section 4. Starting from the observation that non-strict computations with defined results cannot depend on undefined inputs, we derive two additional laws. Using another partial order we develop an equivalent, algebraically elegant form of these properties. All our programming constructs satisfy them, but they are also applicable to relations modelling new constructs.

In short, the contributions of this paper are a new, relational model of imperative, non-deterministic, non-strict computations and a relational description of dependence in such computations.

This paper is a condensed account of a part of the author's PhD thesis [8]. We present the key definitions and results, but omit their proofs. The work grew out of research on Hoare and He's Unifying Theories of Programming [11], however it can be discussed independently and without prior knowledge of that context. The original motivation and many connections to the Unifying Theories of Programming are included in [8].

2 Relational Preliminaries

In this section we set up the context of the investigation of non-strictness. We describe the relational model of imperative, non-deterministic programs in detail and introduce terminology, notation and conventions used in this paper.

Characteristic features of imperative programming are variables, states and statements. We assume an infinite supply $x_1, x_2, \dots$ of variables. Associated with each variable x_i is its type or range D_i , a set comprising all values the variable can take. Each D_i shall contain two special elements $\perp$ and ∞ with the following intuitive meaning: If the variable x_i has the value $\perp$ *and* this value is needed, the execution of the program aborts. If the variable x_i has the value ∞ *and* this value is needed, the execution of the program does not terminate.

A state is given by the values of a finite but unbounded number of variables $x_1, \dots, x_m$ which we abbreviate as $\vec{x}$. Let $1..m$ denote the first m positive integers. The relative complement of a subset $I \subseteq 1..m$ is denoted by $\bar{I} =_{\text{def}} 1..m \setminus I$, where m will be clear from the context. We abbreviate $\{\bar{i}\}$ as $\bar{i}$. Let $\vec{x}_I$ denote the subsequence of $\vec{x}$ comprising those x_i with $i \in I$. By writing $a \in \vec{x}$ or $\vec{x} = a$ we express that $a = x_i$ for some or all $i \in 1..m$, respectively. Let $D_I =_{\text{def}} \prod_{i \in I} D_i$ denote the Cartesian product of the ranges of the variables x_i with $i \in I$. A state is an element $\vec{x} \in D_{1..m}$.

The effect of statements is to transform states into new states. We therefore distinguish the values of a variable x_i before and after the execution of a statement. The input value is denoted just as the variable by x_i and the output value is denoted by x'_i . In particular, both $x_i \in D_i$ and $x'_i \in D_i$. The output state $(x'_1, \dots, x'_n)$ is abbreviated as $\vec{x}'$. Statements may introduce new variables into the state and remove variables from the state; then $m \neq n$.

A computation is modelled as a relation $R = R(\vec{x}, \vec{x}') \subseteq D_{1..m} \times D_{1..n}$. An element $(\vec{x}, \vec{x}') \in R$ intuitively means that the execution of R with input values $\vec{x}$ may yield the output values $\vec{x}'$. The image of a state $\vec{x}$ is given by $R(\vec{x}) =_{\text{def}} \{\vec{x}' \mid (\vec{x}, \vec{x}') \in R\}$. Non-determinism is modelled by having $|R(\vec{x})| > 1$.

Another way to state the type of the relation is $R : D_{1..m} \leftrightarrow D_{1..n}$. The framework employed is that of heterogeneous relation algebra [22, 23]. We omit any notational distinction of the types of relations and their operations and assume type-correctness in their use.

We denote the zero, identity and universal relations by $\perp$, $\mathbb{I}$ and $\mathbb{T}$, respectively. Lattice join, meet and order of relations are denoted by $\cup$, $\cap$ and $\subseteq$, respectively. The Boolean complement of R is $\bar{R}$, and the converse (transposition) of R is $R^\sim$. Relational (sequential) composition of P and Q is denoted by $P ; Q$ and PQ . Converse has highest precedence, followed by sequential composition, followed by meet and join with lowest precedence.

A relation R is a vector iff $R\mathbb{T} = R$, total iff $R\mathbb{T} = \mathbb{T}$ and univalent iff $R^\sim R \subseteq \mathbb{I}$. A relation is a mapping iff it is both total and univalent.

Relational constants representing computations may be specified by set comprehension as, for example, in

$$R = \{(\vec{x}, \vec{x}') \mid x'_1 = x_2 \wedge x'_2 = 1\} = \{(\vec{x}, \vec{x}') \mid x'_1 = x_2\} \cap \{(\vec{x}, \vec{x}') \mid x'_2 = 1\}.$$

We abbreviate such a comprehension by its constituent predicate, that is, we write $R = x'_1 = x_2 \cap x'_2 = 1$. In doing so, we use the identifier x in a generic way, possibly decorated with an index, a prime or an arrow. It follows, for example, that $\vec{x} = \vec{c}$ is a vector for any constant $\vec{c}$. Generally used to construct relational constants, infix operators without spacing have higher precedence than converse.

To form heterogeneous relations and, more generally, to change their dimensions, we use the following projection operation. Let I, J, K and L be index sets such that $I \cap K = \emptyset = J \cap L$. The dimensions of $R : D_{I \cup K} \leftrightarrow D_{J \cup L}$ are restricted by

$$(\exists \vec{x}_K, \vec{x}'_L : R) =_{\text{def}} \{(\vec{x}_I, \vec{x}'_J) \mid \exists \vec{x}_K, \vec{x}'_L : (\vec{x}_{I \cup K}, \vec{x}'_{J \cup L}) \in R\} : D_I \leftrightarrow D_J.$$

We abbreviate the case $L = \emptyset$ as $(\exists \vec{x}_K : R)$ and the case $K = \emptyset$ as $(\exists \vec{x}'_L : R)$.

Defined in terms of the projection, we furthermore use the following relational parallel composition operator, similar to that of [13]. The parallel composition of the relations $P : D_I \leftrightarrow D_J$ and $Q : D_K \leftrightarrow D_L$ is

$$P \parallel Q =_{\text{def}} (\exists \vec{x}'_K : \mathbb{I}) ; P ; (\exists \vec{x}_L : \mathbb{I}) \cap (\exists \vec{x}'_I : \mathbb{I}) ; Q ; (\exists \vec{x}_J : \mathbb{I}) : D_{I \cup K} \leftrightarrow D_{J \cup L}.$$

If necessary, we write $P_I \parallel_K Q$ to clarify the partition of $I \cup K$ (a more detailed notation would also clarify the partition of $J \cup L$). The $\parallel$ operator has lower precedence than meet and join.

The scope of quantifiers in a formula extends as far to the right as possible, that is, until the next unmatched closing bracket or the end of the formula. Logical quantification over the empty sequence of variables can be omitted, that is, $(\exists \vec{x}_\emptyset : A) = (\forall \vec{x}_\emptyset : A) = A$.

3 Programming Constructs

We present a relational model of non-strict computations. In particular, we give new definitions for a number of programming constructs and identify several algebraic properties they satisfy. The latter starts with isotony and the unit laws in Section 3.1, followed by boundedness, continuity and totality in Section 3.2 and two dependence conditions in Section 4.

Basic statements comprise the assignment, skip, (un)declaration of variables and alphabet extension. Control flow is provided by the conditional, sequential and parallel composition. Relations may furthermore be composed by the non-deterministic choice. Its dual, conjunction, is technically useful for the treatment of recursion which is given by the greatest fixpoint. We moreover consider its dual, the least fixpoint. This selection of programming constructs subsumes the imperative, non-deterministic core of the Unifying Theories of Programming [11].

3.1 Isotony and Neutrality

We successively define our programming constructs using relations and discuss essential algebraic properties. At first we introduce a fundamental order on the variable ranges, which is used throughout this paper.

Recall that the range D_i of a variable contains the special elements $\perp$ and ∞ modelling undefinedness and non-termination, respectively. Let $\preceq : D_i \leftrightarrow D_i$ be the flat order on D_i with ∞ as its least element, that is, $x \preceq y \Leftrightarrow_{\text{def}} x = \infty \vee x = y$. It follows that $\preceq$ is a partial order and even a meet-semilattice. A similar order, in which $\perp$ is the least element, will be introduced in Section 4.1.

Recall further that $D_I = \prod_{i \in I} D_i$. Let $\preceq : D_I \leftrightarrow D_I$ also denote the pointwise extension of that order, that is, $\vec{x}_I \preceq \vec{y}_I \Leftrightarrow_{\text{def}} \forall i \in I : x_i \preceq y_i$. Its dual order is denoted by $\succ =_{\text{def}} \preceq^\sim$. The meet operation is obtained by pointwise extension, too. We exclusively work with finite I , indexing the variables of the current state. It is easily proved by induction on the size of the index set I , that $|C| \leq |I| + 1$ for any chain C in D_I ordered by $\preceq$. It follows that the corresponding strict order $\prec$ is regressively bounded and therefore also well-founded.

Most of the time we use the partial order $\preceq$ with the index set $I = 1..m$ of all variables, as in $\vec{x} \preceq \vec{x}'$. Indeed, we take this as the definition of the new relation modelling skip, denoted also by $\mathbb{1} =_{\text{def}} \preceq$. The intention underlying the definition of $\mathbb{1}$ is to enforce an upper closure of the image of each state with respect to $\preceq$. Traces of such a procedure can be found in the healthiness condition H2 of [11] and in the $\perp$ -predicates of [7]. Our definition of $\mathbb{1}$ refines this by distinguishing individual variables. As usual, skip should be a left and right unit of sequential composition.

Definition 1. $\mathcal{H}_L(P) \Leftrightarrow_{\text{def}} \mathbb{1} ; P = P$ and $\mathcal{H}_R(P) \Leftrightarrow_{\text{def}} P ; \mathbb{1} = P$.

By reflexivity of $\mathbb{1}$ it suffices to demand $\subseteq$ instead of equality. We furthermore use $\mathcal{H}_E(P) \Leftrightarrow_{\text{def}} \mathcal{H}_L(P) \wedge \mathcal{H}_R(P)$. It follows that for $X \in \{E, L, R\}$ the relations satisfying $\mathcal{H}_X$ form a complete lattice. The rest of this section is devoted to giving definitions of programming constructs that satisfy or preserve these laws.

The assignment statement is usually defined as the mapping $\vec{x} := \vec{e} =_{\text{def}} \vec{x}' = \vec{e}$, where each expression $e \in \vec{e}$ may depend on the input values $\vec{x}$ of the variables, and yields *exactly one* value $e(\vec{x})$ from the expression's type. Our new relation modelling the assignment is $\vec{x} \leftarrow \vec{e} =_{\text{def}} \mathbb{1} ; \vec{x} := \vec{e} ; \mathbb{1}$. We write $\vec{x} \leftarrow e$ to assign the same expression e to all variables. The upper closure of the images perspicuously appears in the following lemma which intuitively states that $\mathbb{T}$ models the never terminating program.

Lemma 2. We have $\vec{x} \leftarrow \infty = \mathbb{T}$ and $\vec{x} \leftarrow \vec{c} = \vec{x}' = \vec{c} = \vec{x} := \vec{c}$ for any $\vec{c} \in D_{1..n}$ such that $\infty \notin \vec{c}$.

Resuming our introductory example we now obtain $x_1, x_2 \leftarrow \perp, 2 ; x_1 \leftarrow x_2 = x_1, x_2 \leftarrow 2, 2$ and furthermore $\mathbb{T} ; x_1, x_2 \leftarrow 2, 2 = x_1, x_2, \vec{x}_{3..n} \leftarrow 2, 2, \infty$. This demonstrates that computations in our setting are indeed non-strict.

To deal with the conditional and later also with the assignment, we need to restrict the expressions that occur on the right hand side of assignments and as conditions. We assume that the expressions are isotone with respect to $\preceq$ as captured by the following condition.

Definition 3. Let E be a partial order. The sequence of expressions $\vec{e}$ is isotone with respect to E iff $\mathcal{I}_E(\vec{e}) \Leftrightarrow_{\text{def}} E ; \vec{x}' = \vec{e} \subseteq \vec{x}' = \vec{e} ; E$.

At this stage we need $\mathcal{I}_{\preceq}(\vec{e})$, that is, $\preceq ; \vec{x}' = \vec{e} \subseteq \vec{x}' = \vec{e} ; \preceq$. If the expression e is viewed as a function, then $\mathcal{I}_{\preceq}(e)$ amounts to the usual isotony in partially ordered sets, namely $\forall \vec{x}, \vec{y} : \vec{x} \preceq \vec{y} \Rightarrow e(\vec{x}) \preceq e(\vec{y})$. Its relational formulation appears, for example, in [21]. It can be shown that any expression composed of constants, variables and strict functions is isotone, thus the restriction is not too severe.

Let us elaborate the assignment $\vec{x} \leftarrow \vec{e}$ assuming $\mathcal{I}_{\preceq}(\vec{e})$. It then simplifies to $\vec{x} \leftarrow \vec{e} = \vec{x} := \vec{e} ; \mathbb{1}$ since $\mathbb{1} ; \vec{x}' = \vec{e} ; \mathbb{1} \subseteq \vec{x}' = \vec{e} ; \mathbb{1} ; \mathbb{1} = \vec{x}' = \vec{e} ; \mathbb{1} \subseteq \mathbb{1} ; \vec{x}' = \vec{e} ; \mathbb{1}$. Hence $\vec{x} \leftarrow \vec{e} = \vec{x}' = \vec{e} ; \mathbb{1} = \{(\vec{x}, \vec{x}') \mid \exists \vec{y} : \vec{y} = \vec{e}(\vec{x}) \wedge \vec{y} \preceq \vec{x}'\} = \{(\vec{x}, \vec{x}') \mid \vec{e}(\vec{x}) \preceq \vec{x}'\}$. This means that the successor states of $\vec{x}$ under this assignment comprise the usual successor $\vec{e}(\vec{x})$ and its upper closure with respect to $\preceq$.

We treat conditions as expressions with values in $\{\infty, \perp, \text{true}, \text{false}\}$ that may depend on the input $\vec{x}$. If b is a condition, the relation $b=c$ is a vector for any $c \in \{\infty, \perp, \text{true}, \text{false}\}$. Recalling how relational constants are specified, and using $\vec{x}_{1..m}$ as input variables, we obtain that $b=c = \{(\vec{x}, \vec{x}') \mid b(\vec{x})=c\} : D_{1..m} \leftrightarrow D_{1..n}$ for arbitrary $D_{1..n}$ depending on the context. The new relation modelling the conditional ‘if b then P else Q ’ is

$$(P \blacktriangleleft b \blacktriangleright Q) =_{\text{def}} b=\infty \cup (b=\perp \cap \vec{x}'=\perp) \cup (b=\text{true} \cap P) \cup (b=\text{false} \cap Q).$$

The effect of an undefined condition in a conditional statement is to set all variables of the current state undefined. By Lemma 2 we can indeed replace $b=\infty \cup (b=\perp \cap \vec{x}'=\perp)$ with $(b=\infty \cap \vec{x} \leftarrow \infty) \cup (b=\perp \cap \vec{x} \leftarrow \perp)$. This models the fact that the evaluation of b is always necessary if the execution of the conditional is. Any non-termination or undefinedness is thus propagated.

Variables are added to and removed from the current state by the projection operators. We adapt them to respect $\mathcal{H}_E$; our relations modelling variable (un)declaration are **var** $\vec{x}_K =_{\text{def}} (\exists \vec{x}_K : \mathbb{1})$ and **end** $\vec{x}_K =_{\text{def}} (\exists \vec{x}'_K : \mathbb{1})$. At this place, inhomogeneous relations enter the stage. The basic declaration can be augmented to provide initialised variable declarations.

To hide local variables from recursive calls [11] uses the *alphabet extension*. We generalise it to handle several variables and heterogeneous relations. Let $P : D_I \leftrightarrow D_J$, then our alphabet extension is $P^{+\vec{x}_K} : D_{I \cup K} \leftrightarrow D_{J \cup K}$ given by

$$P^{+\vec{x}_K} =_{\text{def}} \text{end } \vec{x}_I ; \text{var } \vec{x}_J \cap \text{end } \vec{x}_K ; P ; \text{var } \vec{x}_K.$$

Intuitively, the part **end** $\vec{x}_I ; \text{var } \vec{x}_J$ preserves the values of $\vec{x}_K$ and the part **end** $\vec{x}_K ; P ; \text{var } \vec{x}_K$ applies P to $\vec{x}_I$ to obtain $\vec{x}_J$. Just as the variable undeclaration may be seen as a projection, the alphabet extension is an instance of relational parallel composition. This follows since $P^{+\vec{x}_K} = \mathbb{1} P \mathbb{1}_{I \parallel K} \mathbb{1}$, which simplifies to $P_{I \parallel K} \mathbb{1}$ if $\mathcal{H}_E(P)$ holds. It is typically as complex to prove a result for the more general $P \parallel Q$ as it is for $P^{+\vec{x}_K}$; we therefore use the former.

We have now introduced a selection of programming constructs as summarised in the following definition. This selection is inspired by [11] and rich enough to yield a basic programming and specification language.

Definition 4. We use the following relations and operations:

<i>skip</i>	$\mathbb{1} =_{\text{def}} \prec$
<i>assignment</i>	$\vec{x} \leftarrow \vec{e} =_{\text{def}} \mathbb{1} ; \vec{x} := \vec{e} ; \mathbb{1}$
<i>variable declaration</i>	var $\vec{x}_K =_{\text{def}} (\exists \vec{x}_K : \mathbb{1})$
<i>variable undeclaration</i>	end $\vec{x}_K =_{\text{def}} (\exists \vec{x}'_K : \mathbb{1})$
<i>parallel composition</i>	$P \parallel Q$
<i>sequential composition</i>	$P ; Q$
<i>conditional</i>	$(P \blacktriangleleft b \blacktriangleright Q) =_{\text{def}} b=\infty \cup (b=\perp \cap \vec{x}'=\perp) \cup (b=\text{true} \cap P) \cup (b=\text{false} \cap Q)$
<i>non-deterministic choice</i>	$\bigcup_{P \in S} P$
<i>conjunction</i>	$\bigcap_{P \in S} P$
<i>greatest fixpoint</i>	$\nu f =_{\text{def}} \bigcup \{P \mid f(P) = P\}$
<i>least fixpoint</i>	$\mu f =_{\text{def}} \bigcap \{P \mid f(P) = P\}$

Composition, choice and fixpoint are just the familiar operations of relation algebra. This simplifies reasoning because it enables applying familiar laws, like distribution of $;$ over $\cup$, also to programs. We use the *greatest* fixpoint to define the semantics of specifications given by recursive equations and thus obtain demonic non-determinism. For example, the iteration **while** b **do** P is just $\nu(\lambda X.P ; X \blacktriangleleft b \blacktriangleright \mathbb{1})$.

We conclude our compendium of programming constructs by two useful results. The first states isotony, which is important for the existence of fixpoints needed to solve recursive equations. The second establishes $\mathbb{1}$ as a left and right unit of sequential composition, which is useful to terminate iterations and to obtain a one-sided conditional. Necessary restrictions of the theorems in this paper are summarised in Table 11 in Section 5.

Theorem 5. *Functions composed of the constructs of Definition 4 with the restrictions stated in Table 7 are isotone.*

Theorem 6. *Relations composed of the constructs of Definition 4 with the restrictions stated in Table 7 satisfy $\mathcal{H}_R$ and $\mathcal{H}_L$. The latter requires $\mathcal{T}_{\preceq}(b)$ for all conditions b .*

3.2 Finite Branching

From the computational perspective, it is necessary to regard the greatest fixpoint not as the supremum of all fixpoints but as the infimum of a certain chain. Not all properties, however, are preserved by infima of chains. It occasionally helps to restrict the attention to infima of chains of relations that model a finite degree of non-determinism. Such relations represent what are sometimes called *boundedly non-deterministic* programs, see [6,10,27]. In graph theory, taking states as nodes and transitions as edges, one speaks of a finite outdegree. As elaborated below, the pure condition of finite branching is not appropriate. We therefore provide a new, relaxed condition. Finite branching is necessary to show the continuity of functions and the totality of relations, which we do afterwards.

To prepare our definition of finite branching, we have to discuss minimal elements of the set $D_{1..n}$ ordered by $\preceq$. Since many results also hold in more general orders, we abstract to a set S partially ordered by $\preceq$. The *minimal elements* of $A \subseteq S$ are $\min A =_{\text{def}} \{x \mid x \in A \wedge \forall y : (y \in A \wedge y \preceq x) \Rightarrow y = x\}$. We call S *well-founded* iff $\min A \neq \emptyset$ for all $\emptyset \neq A \subseteq S$. The *upper closure* of $A \subseteq S$ is $\uparrow A =_{\text{def}} \{y \mid y \in S \wedge \exists x \in A : x \preceq y\}$ and A is an *upper set* iff $A = \uparrow A$.

These concepts are connected to computations by applying them to the image set of each state with $\preceq$ as the partial order. We have already observed that $D_{1..n}$ is well-founded and the following lemma establishes these images as upper sets provided the computation satisfies $\mathcal{H}_R$.

Lemma 7. *If S is well-founded and $A \subseteq S$ is an upper set, then $A = \uparrow \min A$. Furthermore, $\mathcal{H}_R(P)$ holds for a relation P iff $P(\vec{x})$ is an upper set for all $\vec{x}$.*

This provides the link between the relation-algebraic viewpoint of $\mathcal{H}_R$ and the pointwise upper sets. One can represent and calculate with minima as relations, see [23] and Section 4.3, but the proof of Lemma 7 remains essentially pointwise.

We are ready to state the condition for boundedly non-deterministic computations. Traditional finite branching cannot be used since we need $\top$ to represent the never terminating program. This is due to the demonic interpretation of non-deterministic choice. The condition that each state $\vec{x}$ has only a finite number of successor states can be relaxed by allowing additionally the case that every state in $D_{1..n}$ is a successor of $\vec{x}$ [10]. This solves the problem with $\top$, that satisfies the relaxed condition, but is not fine enough for our purposes. We further need to distinguish the individual variables, which is done by the condition $\mathcal{H}_B$ using the pointwise minima with respect to $\preceq$.

Definition 8. $\mathcal{H}_B(P) \Leftrightarrow_{\text{def}} \forall \vec{x} : |\min P(\vec{x})| \in \mathbb{N}$.

The intention of using $\min$ is the following: $\mathcal{H}_B$ will be applied to relations that satisfy $\mathcal{H}_R$. By Lemma 7 the image sets of such relations are in a one-to-one correspondence with their minimal elements. Indeed, it is the minimal elements that actually represent the successor states, and their upper closure is formed to satisfy $\mathcal{H}_R$ and to avoid unboundedness. Thus $\mathcal{H}_B$ accounts for the proper successor states, excluding those that have been added for technical reasons. We can show that many relations from our compendium satisfy $\mathcal{H}_B$.

Theorem 9. *Relations composed of the constructs of Definition 4 with the restrictions stated in Table 1 satisfy $\mathcal{H}_B$. In particular, $\mathcal{I}_{\preceq}(\vec{e})$ is required for all expressions $\vec{e}$.*

The proof uses the fact that $D_{1..n}$ ordered by $\preceq$ is a meet-semilattice having finite height. Finite height (which implies well-foundedness) is guaranteed since there are only a finite number of variables and the ranges D_i are flat orders. The latter suffices for data structures with strict constructors, but excludes infinite data structures which are modelled by non-flat orders. However, the problem is not caused by the infinite data structures themselves, but by having non-determinism at the same time. A more general investigation using powerdomains with finitely generable elements [18,20,26] is postponed to future work.

We call a function f *continuous* iff it distributes over infima of non-empty chains of relations, formally $f(\bigcap C) = \bigcap_{P \in C} f(P)$ for each chain $C \neq \emptyset$. The importance of continuity comes from the permission to represent the greatest fix-point νf by the constructive $\bigcap_{n \in \mathbb{N}} f^n(\top)$. This enables the approximation of νf by repeatedly unfolding f , which simulates recursive calls of the modelled computation. That infinite branching or unbounded non-determinism breaks continuity is shown, for example, in [6, Chapter 9] and [5, Section 5.7]. We use the finite branching property $\mathcal{H}_B$ to establish the continuity of functions composed in our framework.

Theorem 10. *Functions composed of the constructs of Definition 4 with the restrictions stated in Table 1 are continuous, that is, distribute over infima of non-empty chains of relations satisfying $\mathcal{H}_E$ and $\mathcal{H}_B$.*

The proof uses the following two distribution results.

1. Let C be a non-empty chain such that $\mathcal{H}_R(P)$ and $\mathcal{H}_B(P)$ for all $P \in C$. Then $(\bigcap_{P \in C} PQ) = (\bigcap C)Q$.
2. Let C be a non-empty chain such that $\mathcal{H}_L(Q)$ for all $Q \in C$, and let P be such that $\mathcal{H}_R(P)$ and $\mathcal{H}_B(P)$. Then $(\bigcap_{Q \in C} PQ) = P(\bigcap C)$.

Besides finite branching, another reasonable condition for computation purposes is totality, or non-empty branching. Consider the usual interpretation of relations as programs and specifications. Then $\subseteq$ models refinement: $P \subseteq Q$ states that the program P implements the specification Q , because any observation of the execution of P is admitted by Q . But since the empty relation $\perp$ is the least element with respect to $\subseteq$, it implements *any* specification. More generally, the refinement interpretation of P fails if some state has no successors under P . This is prevented by requiring totality of relations.

Theorem 11. *Let $\mathcal{H}_T(P) \Leftrightarrow_{\text{def}} P ; \top = \top$. Relations composed of the constructs of Definition 4 with the restrictions stated in Table 1 satisfy $\mathcal{H}_T$.*

4 Dependence

We now have a relational theory of computations where undefined and defined variables coexist. In this section we discuss two aspects of non-strictness that can be described in terms of dependence of variables. The first gives conditions in case the computation has non-strict parts, and the second gives conditions if it has no strict parts. Let us illustrate the distinction in the case $m = n = 1$, that is, a single input and output variable.

The relation R has a non-strict part if there is an $x'_1 \neq \perp$ such that $(\perp, x'_1) \in R$. For this part, the value of x'_1 must not depend on the value of x_1 or else the input $x_1 = \perp$ would result in the output $x'_1 = \perp$. In other words, there must be a constant assignment to x'_1 . We therefore obtain the condition $(x_1, x'_1) \in R$ for all x_1 . This essentially reflects that one cannot test for undefinedness: If the value of a variable is undefined, such a test is undefined, too.

The relation R has no strict part if $(\perp, \perp) \notin R$. Then the value of x'_1 must not depend on the value of x_1 for any part. Hence the above condition is not sufficient because we must assure that *only* constant assignments occur. This is achieved by requiring $(x_1, x'_1) \in R$ for all x_1 , if $(x_1, x'_1) \in R$ for some x_1 . Note that choosing $x_1 = \perp$ yields a special case of the first condition, while $x'_1 = \perp$ is prevented since it implies $(\perp, \perp) \in R$.

In the following two sections, each of these conditions is generalised to arbitrary m and n , then expressed relationally and in order-theoretic terms, and finally applied to our programming constructs.

For a sequence $\vec{x}$ of length n let $\vec{x}_{i \rightarrow a}$ denote $x_1, \dots, x_{i-1}, a, x_{i+1}, \dots, x_n$, that is, the replacement of x_i by a . If $I \subseteq 1..n$, let $\vec{x}_{I \rightarrow a}$ denote the replacement of x_i by a in $\vec{x}$ for all $i \in I$.

4.1 Non-strict Parts

We first deal with the non-strict parts of a relation. Let us formalise the case $m = n = 1$. As stated above, a non-strict part of the relation R is given by an outcome $x'_1 \neq \perp$ for $x_1 = \perp$. Then x'_1 must be an outcome for all x_1 . We thus have

$$\forall x'_1 : (x'_1 \neq \perp \wedge (\perp, x'_1) \in R) \Rightarrow \forall x_1 : (x_1, x'_1) \in R.$$

By a series of generalisations we obtain the following predicate for arbitrary m and n (choose $m = n = i = 1$ and $J = \emptyset$ to recover the special case, observing that $\vec{i} = \emptyset$ and $\vec{J} = \{1\}$ and $(\vec{x}_{i \mapsto \perp}, \vec{x}'_{J \mapsto \perp}) = (\perp, x'_1)$ hold):

$$\begin{aligned} \forall i \in 1..m : \forall J \subseteq 1..n : \forall \vec{x}_{\vec{i}} : \forall \vec{x}'_{\vec{J}} : \\ (\perp \notin \vec{x}'_{\vec{J}} \wedge (\vec{x}_{i \mapsto \perp}, \vec{x}'_{J \mapsto \perp}) \in R) \Rightarrow \forall x_i : \exists \vec{x}'_{\vec{J}} : (\vec{x}, \vec{x}') \in R. \end{aligned}$$

Intuitively, the antecedent states that for $x_i = \perp$ there is an outcome such that $x'_j \neq \perp$ if and only if $j \notin J$. Then all such x'_j must not depend on x_i . This means that there must be an outcome with these values of x'_j for all values of x_i . The general condition can be equivalently transformed into relational terms:

$$\forall i \in 1..m : \forall J \subseteq 1..n : x_i := \perp ; R \cap \vec{x}_J = \perp \subseteq R ; \vec{x}_J := \perp \cup \perp \in \vec{x}'_{\vec{J}}. \quad (1)$$

We can also derive an order-theoretic representation of (1). To this end, we introduce an order similar to $\preceq$, but now with respect to $\perp$. Let $\sqsubseteq : D_i \leftrightarrow D_i$ be the flat order on D_i with $\perp$ as its least element, that is, $x \sqsubseteq y \Leftrightarrow_{\text{def}} x = \perp \vee x = y$. Again, the order is extended pointwise to D_I by $\vec{x}_I \sqsubseteq \vec{y}_I \Leftrightarrow_{\text{def}} \forall i \in I : x_i \sqsubseteq y_i$, and its dual order is denoted by $\sqsupseteq =_{\text{def}} \sqsubseteq^\sim$. The properties of $\preceq$ can be transferred to $\sqsubseteq$. Using this order, we obtain an algebraic characterisation.

Lemma 12. *Let $\mathcal{H}_N(R) \Leftrightarrow_{\text{def}} \sqsupseteq ; R \subseteq R ; \sqsubseteq$. Then (1) $\Leftrightarrow \mathcal{H}_N(R)$.*

If R is a mapping, the condition $\mathcal{H}_N(R)$ states that R is isotone with respect to $\sqsubseteq$ [21]. Further remarks about $\mathcal{H}_N$ are given in Section 4.2 once the second condition is established. Let us emphasise that $\sqsubseteq$ serves to support our reasoning about undefinedness, that is, finite failure. It is not used to approximate fixpoints, which we do by the subset order $\subseteq$ that (with closure under $\preceq$) corresponds to an order based on wp. In [16] two orders based on wp and wlp are combined for approximation.

We can show that our programming constructs satisfy $\mathcal{H}_N$. To deal with the assignment and the conditional, we assume that the expressions are isotone with respect to $\sqsubseteq$. The proof of the following result requires $\mathcal{T}_{\preceq}(\vec{e})$ and $\mathcal{T}_{\sqsubseteq}(\vec{e})$ for all expressions $\vec{e}$. Since $\preceq$ and $\sqsubseteq$ are structurally similar, the properties of $\mathcal{T}_{\preceq}$ can be transferred to $\mathcal{T}_{\sqsubseteq}$.

Theorem 13. *Relations composed of the constructs of Definition 4 with the restrictions stated in Table 1 satisfy $\mathcal{H}_N$.*

4.2 Absent Strict Parts

We now treat the case where relations have no strict parts. Let us again start with formalising the case $m = n = 1$. As stated at the beginning of Section 4, the relation R has no strict part if $(\perp, \perp) \notin R$. We then must make sure that the value of x'_1 does not depend on the value of x_1 . In other words, any outcome x'_1 must be an outcome for all x_1 . We therefore have

$$(\perp, \perp) \notin R \Rightarrow \forall x_1, x'_1 : (x_1, x'_1) \in R \Rightarrow \forall \tilde{x}_1 : (\tilde{x}_1, x'_1) \in R.$$

By a series of generalisations we obtain the following predicate for arbitrary m and n (choose $m = n = i = j = 1$ to recover the special case, observing that $\vec{i} = \vec{j} = \emptyset$ and $(\vec{x}_{i \rightarrow \perp}, \vec{x}'_{j \rightarrow \perp}) = (\perp, \perp)$ and $(\vec{x}_{i \rightarrow \tilde{x}_i}, \vec{x}'_{j \rightarrow x'_j}) = (\tilde{x}_1, x'_1)$ hold):

$$\begin{aligned} \forall i \in 1..m : \forall j \in 1..n : \forall \vec{x}'_j : (\forall \vec{x}_j : (\vec{x}_{i \rightarrow \perp}, \vec{x}'_{j \rightarrow \perp}) \notin R) \Rightarrow \\ \forall x_i : \forall \vec{x}' : (\vec{x}, \vec{x}') \in R \Rightarrow \forall \tilde{x}_i : \exists \vec{x}'_j : (\vec{x}_{i \rightarrow \tilde{x}_i}, \vec{x}'_{j \rightarrow x'_j}) \in R. \end{aligned}$$

Intuitively, the first antecedent states that for $x_i = \perp$ there is no outcome such that $x'_j = \perp$. Then x'_j must not depend on x_i . This means that if there is an outcome $\vec{x}'$ for some value of x_i , there must be an outcome with the same value of x'_j for all values of x_i . We can again equivalently transform to relational terms:

$$\forall i \in 1..m : \forall j \in 1..n : \vec{x}_i = \vec{x}'_i ; R \subseteq x_i := \perp ; R ; x_j := \perp \cup R ; x_j = x'_j. \quad (2)$$

It turns out that we have to strengthen this condition to be able to prove closure under sequential composition. The reason is that the two occurrences of R on the right hand side are not coupled tightly enough. Such a problem did not arise with $\mathcal{H}_N$ that is structurally simpler, but it is solved in Lemma 14. Using the order $\sqsubseteq$ introduced in Section 4.1, we can derive an algebraic characterisation. It is proved to be stronger than (2) in the presence of $\mathcal{H}_N$.

Lemma 14. *Let $\mathcal{H}_A(R) \Leftrightarrow_{\text{def}} \sqsubseteq ; R \subseteq R ; \sqsubseteq$ and consider*

$$\forall I \subseteq 1..m : \vec{x}_I = \vec{x}'_I ; R \subseteq \bigcup_{J \subseteq 1..n} \vec{x}_I := \perp ; R ; \vec{x}_J := \perp^\sim \cap R ; \vec{x}_J = \vec{x}'_J. \quad (3)$$

Then (2) $\Leftarrow$ (3) $\Rightarrow \mathcal{H}_A(R)$. If $\mathcal{H}_N(R)$ holds, then (3) $\Leftrightarrow \mathcal{H}_A(R)$.

This lemma suggests to use the conjunction of $\mathcal{H}_A$ and $\mathcal{H}_N$ since it is equivalent to a stronger form of the derived conditions. If R is a mapping, we have $\mathcal{H}_N(R) \Leftrightarrow \mathcal{H}_A(R)$. But also the other programming constructs satisfy $\mathcal{H}_A$.

Theorem 15. *Relations composed of the constructs of Definition 4 with the restrictions stated in Table 1 satisfy $\mathcal{H}_A$.*

A general form of the conditions $\mathcal{H}_N$ and $\mathcal{H}_A$ appears in the literature, although in another context and not in relational form. Let $E : A \leftrightarrow A$ be a partial order and $R : A \leftrightarrow A$ a relation that satisfies $ER \subseteq RE$ and $E^\sim R \subseteq RE^\sim$. Then R

is called an *isotone relation* [28] and an *order preserving multifunction* [25]. In both cases, the definition is given pointwise, requiring for all $(x_1, x_2) \in E$ that

- for each $y_1 \in R(x_1)$ there is a $y_2 \in R(x_2)$ such that $(y_1, y_2) \in E$, and
- for each $y_2 \in R(x_2)$ there is a $y_1 \in R(x_1)$ such that $(y_1, y_2) \in E$.

The investigation is concerned with the question whether A ordered by E satisfies the *relational fixed point property* [24]. This is the case iff every total, isotone relation R has a fixed point $x \in A$ such that $x \in R(x)$. Such a study has the relations themselves, interpreted as orders, as its objects. This has to be contrasted with our effort to obtain fixpoints of isotone functions over relations.

The two criteria stated above express precisely what constitutes the Egli-Milner order on powerdomains built from flat domains [18,20]. One can interpret the conjunction of $\mathcal{H}_N$ and $\mathcal{H}_A$ as imposing the Egli-Milner order on the image sets of relations. This order is frequently used in semantics but in different ways and for a different purpose. For example, in [24] it orders relations, while in [5,9,27] it orders domains of functional programming languages. All these sources use the Egli-Milner order to define the least fixpoint of functions. In our approach, however, fixpoints are ordered by the usual subset relation and the Egli-Milner order appears merely in the conditions $\mathcal{H}_N$ and $\mathcal{H}_A$ dealing with undefinedness. As a matter of fact the Egli-Milner order models erratic non-determinism or general correctness, but our definitions model demonic non-determinism or total correctness; see [16,27] for the difference.

The conditions $\mathcal{H}_N$ and $\mathcal{H}_A$ can also be seen as expressing an information preservation principle. In this interpretation $\sqsubseteq$ is the definedness information order and $\mathcal{H}_N$ and $\mathcal{H}_A$ convey definedness information. Corresponding conditions for the termination information order $\preceq$ are discussed in Section 4.3. This view fits well with the notion of *partiality* investigated in [21]: ‘A treatment of possibly partial availability of information may also be seen in descriptions of eager/data-driven evaluation as opposed to lazy/demand-driven evaluation.’[ibid., page 213]

4.3 Undefinedness and Non-termination

The conditions $\mathcal{H}_N$ and $\mathcal{H}_A$ introduced in the previous sections model dependence of undefined values. This manifests itself in the use of $\sqsubseteq$ with its least element $\perp$ in their definitions. It is legitimate to ask whether analogous conditions using $\preceq$ with its least element ∞ also hold. More generally, we should elaborate on the relationship between $\sqsubseteq$ and $\preceq$. Although these orders are structurally very similar and thus share several properties, there is an essential difference in their use. It is expressed by the conditions $\mathcal{H}_L$ and $\mathcal{H}_R$ enforcing closure with respect to $\preceq$. The reason why $\preceq$ is used for closure is the chosen model of the never terminating program: This relation should be both $\vec{x} \leftarrow \infty$ and the solution of the recursive equation $X = X$, that is, $\nu(\lambda X.X) = \top$. We thus obtain the requirement $\vec{x} \leftarrow \infty = \top$ which is satisfied by upper closure as Lemma 2 shows. To achieve this closure, $\preceq$ is inherent in our programming constructs. A similar upper closure with respect to $\sqsubseteq$ is neither necessary nor advisable for this would identify non-termination with undefinedness.

This explains why we use conditions of type $\mathcal{H}_L$ and $\mathcal{H}_R$ with respect to $\preceq$ but not $\sqsubseteq$. Let us return to the question of using $\mathcal{H}_N$ - and $\mathcal{H}_A$ -type conditions also with respect to $\preceq$. Such conditions can indeed be stated but we must take into account that the relations are $\mathcal{H}_L$ - and $\mathcal{H}_R$ -closed. Otherwise, simply requiring $\succ ; R \subseteq R ; \succ$ does not work since already for $R = \mathbf{1} = \preceq$ we would obtain $\top = \succ ; \preceq \subseteq \preceq ; \succ$ which does not hold in general. Instead, we have to undo the effects of the upper closure and state conditions analogous to $\mathcal{H}_N$ and $\mathcal{H}_A$ using the minimal elements of the images as in Section 3.2. We use the relational formulation of min, similar to a construction in [23, page 43].

Theorem 16. *Let $\min R =_{\text{def}} R \cap \overline{R \prec}$ and*

$$\begin{aligned}\mathcal{H}_W(R) &\Leftrightarrow_{\text{def}} \preceq ; \min R \subseteq R ; \preceq, \\ \mathcal{H}_M(R) &\Leftrightarrow_{\text{def}} \succ ; \min R \subseteq R ; \succ.\end{aligned}$$

Relations composed of the constructs of Definition 4 with the restrictions stated in Table 7 satisfy $\mathcal{H}_M$ and $\mathcal{H}_W$.

5 Summary and Adequacy

Table 1 summarises the closure properties of the conditions investigated in this paper. It lists for each condition $\mathcal{H}$ those constructs that are allowed in the construction of a relation or function R such that $\mathcal{H}(R)$ can be shown. The column $\exists$ refers to skip and (un)declaration, and the following columns refer to assignment, arbitrary constant relations, parallel composition, sequential composition, conditional, non-deterministic choice, conjunction, greatest and least fixpoints, in this sequence.

An entry $\_$ means that construct is permitted unconditionally. An entry $\mathcal{I}_S$ means $\mathcal{I}_X(\vec{e})$ or $\mathcal{I}_X(b)$ must hold for all $X \in S$. An entry $\mathcal{H}_S$ means the constant must satisfy $\mathcal{H}_X$ for all $X \in S$. An entry $\cup$ means only finite choice is allowed,

Table 1. Closure properties

Theorem	$\exists$	$\leftarrow \vec{e}$	constant	$\parallel$	$;$	$\blacktriangleleft b \blacktriangleright$	$\cup$	$\cap$	ν	μ
5: isotony	-	-	-	-	-	-	-	-	-	-
6: $\mathcal{H}_R$	-	-	$\mathcal{H}_R$	-	-	-	-	-	-	-
6: $\mathcal{H}_L$	-	-	$\mathcal{H}_L$	-	-	$\mathcal{I}_{\preceq}$	-	-	-	-
6: $\mathcal{H}_E$	-	-	$\mathcal{H}_E$	-	-	$\mathcal{I}_{\preceq}$	-	-	-	-
9: $\mathcal{H}_B$	-	$\mathcal{I}_{\preceq}$	$\mathcal{H}_{EB}$	-	-	$\mathcal{I}_{\preceq}$	$\cup$	-	-	$\times$
10: continuity	-	$\mathcal{I}_{\preceq}$	$\mathcal{H}_{EB}$	-	-	$\mathcal{I}_{\preceq}$	$\cup$	-	-	$\times$
11: $\mathcal{H}_T$	-	$\mathcal{I}_{\preceq}$	$\mathcal{H}_{EBT}$	-	-	$\mathcal{I}_{\preceq}$	$\oplus$	$\setminus$	-	$\times$
13: $\mathcal{H}_N$	-	$\mathcal{I}_{\preceq \sqsubseteq}$	$\mathcal{H}_{EBTN}$	-	-	$\mathcal{I}_{\preceq \sqsubseteq}$	$\oplus$	$\setminus$	-	$\times$
15: $\mathcal{H}_A$	-	$\mathcal{I}_{\preceq \sqsubseteq}$	$\mathcal{H}_{EBA}$	-	-	$\mathcal{I}_{\preceq \sqsubseteq}$	$\cup$	$\setminus$	-	$\times$
16: $\mathcal{H}_M$	-	$\mathcal{I}_{\preceq}$	$\mathcal{H}_{EBTM}$	-	-	$\mathcal{I}_{\preceq}$	$\oplus$	$\setminus$	-	$\times$
16: $\mathcal{H}_W$	-	-	$\mathcal{H}_L$	-	-	$\mathcal{I}_{\preceq}$	-	-	-	-

and $\oplus$ requires finite non-empty choice. An entry $\}$ means only chains are allowed. Finally, an entry $\times$ means that construct is not permitted.

We thus obtain a theory similar to [11] but modelling non-strict computations. In particular, the left and right unit laws $\mathcal{H}_L$ and $\mathcal{H}_R$ and the right zero law $\mathcal{H}_T$ correspond to the healthiness conditions H1–H4 without the left zero law $\top ; R = \top$. Moreover, all functions composed of programming constructs are continuous and all relations composed of programming constructs are boundedly non-deterministic. Additionally, they satisfy the conditions $\mathcal{H}_N$ and $\mathcal{H}_A$ modelling the dependence of variables. There is also a correspondence between the constructs introduced in Definition 4 and those of [11] stating that both yield the same results except that our model has better termination properties.

One can furthermore define a formal operational semantics to describe the execution of programs modelled by our constructs. Intuitively, we start with a set of variables whose final values we are interested in, and the execution proceeds backwards, evaluating only those parts actually needed to obtain the required values. Execution of assignments considers the dependences, execution of a conditional evaluates the condition first, and execution of a recursion starts by unfolding. Neither an undefined value nor a non-terminating part has an effect if it is not reached. It follows that our theory models non-strict computations.

6 Conclusion

We have proposed a new relational approach to define the semantics of imperative, non-deterministic programs. Let us summarise its key properties, which also differentiate our theory from related approaches such as [11].

- Undefinedness and non-termination are treated independently of each other. Finite and infinite failure can thus be distinguished which is closer to practice and allows one to model recovery from errors. A fine distinction is offered by dealing with undefinedness separately for individual variables.
- The theory provides a relational model of dependence in computations. Additional laws of programs are stated in a compact algebraic form and can therefore be applied to new programs given as relations.
- The framework supports an operator for the parallel composition of relations. It is used to treat local variable declarations and alphabet extension adequately also in the context of non-termination. Relation algebra is used whenever possible for clear and concise arguments.
- The relations model non-strict computations in an imperative context. Efficiency can thus be improved by executing only those parts of programs necessary to obtain the final results. The theory can serve as a basis to link to the semantics of functional programming languages.

The disadvantages of a possibly lazy evaluation are of course a potential overhead and reduced predictability of execution time, space and order.

Connections to related work have been pointed out throughout this paper. The following description of further approaches is primarily focused on the similarities and differences to the present work.

Undefinedness and non-termination are addressed by [29] using the Z notation. The former is represented by a distinguished element $\perp$ that is propagated through sequential composition and thus models strict computations. Termination is treated by pairs of predicates describing pre- and postconditions. A combination of both aspects is not examined. The Z notation itself does neither deal with undefinedness nor with termination issues [12].

Instead of modelling non-strict computations in an imperative programming language, one can proceed the other way around and introduce state into a lazy functional programming language. A restricted form of state are variables which can be assigned only once as, for example, in [13]. Mutable state is provided by the Haskell I/O monad used in our introductory example. It has the property that all actions are forced, regardless of their contribution to the final result [14][17]. This is avoided using the more general *state transformers* [15], combining lazy evaluation with stateful computation. Since the base language is functional, the semantics is given in the λ -calculus passing around environments and states. For our imperative context this is less adequate as using relations. Non-determinism is not treated and there is no distinction between undefinedness and non-termination.

A multi-paradigm language that supports lazy functions, exception handling, mutable state and non-deterministic choice points is Oz [19]. That book gives a formal operational semantics of the kernel language which, however, does not cover non-determinism. According to the reduction rule for sequential composition, the execution of statements is forced similar to the Haskell I/O monad.

Let us point out a few topics that deserve to be further investigated. One of them concerns the implementation of the presented theory. This involves a deeper study of the operational semantics and its connection to the relational model. Another thread is to explore the relational model as an intermediate for the translation of functional programming languages. The latter should be accompanied by comparing the semantics of lazy evaluation in both frameworks. A different domain is touched by applying the presented model of dependence in computations to develop optimising transformations used, for example, in compilers. Connections are anticipated to abstract interpretation and data flow analysis, where the partial availability of information also plays a role.

Acknowledgements. I am grateful to the anonymous referees for their helpful remarks and thank Bernhard Möller for his detailed comments about [8].

References

1. Backhouse, R.C., de Bruin, P.J., Hoogendijk, P., Malcolm, G., Voermans, E., van der Woude, J.: Polynomial relators (extended abstract). In: Nivat, M., Rattray, C., Rus, T., Scollo, G. (eds.) *Algebraic Methodology and Software Technology*, pp. 303–326. Springer, Heidelberg (1992)
2. de Bakker, J.W.: Semantics and termination of nondeterministic recursive programs. In: Michaelson, S., Milner, R. (eds.) *Automata, Languages and Programming: Third International Colloquium*, pp. 435–477. Edinburgh University Press (1976)

3. Berghammer, R., von Karger, B.: Relational semantics of functional programs. In: Brink, C., Kahl, W., Schmidt, G. (eds.) *Relational Methods in Computer Science*, ch. 8, pp. 115–130. Springer, Wien (1997)
4. Berghammer, R., Zierer, H.: Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science* 43, 123–147 (1986)
5. Broy, M., Gnatz, R., Wirsing, M.: Semantics of nondeterministic and noncontinuous constructs. In: Bauer, F.L., Broy, M. (eds.) *Program Construction*. LNCS, vol. 69, pp. 553–592. Springer, Heidelberg (1979)
6. Dijkstra, E.W.: *A Discipline of Programming*. Prentice-Hall, Englewood Cliffs (1976)
7. Guttman, W.: Non-termination in Unifying Theories of Programming. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) *RelMiCS 2005*. LNCS, vol. 3929, pp. 108–120. Springer, Heidelberg (2006)
8. Guttman, W.: *Algebraic Foundations of the Unifying Theories of Programming*. PhD thesis, Universität Ulm (December 2007)
9. Hennessy, M., Ashcroft, E.A.: The semantics of nondeterminism. In: Michaelson, S., Milner, R. (eds.) *Automata, Languages and Programming: Third International Colloquium*, pp. 478–493. Edinburgh University Press (1976)
10. Hesselink, W.H.: *Programs, Recursion and Unbounded Choice*. Cambridge University Press, Cambridge (1992)
11. Hoare, C.A.R., He, J.: *Unifying theories of programming*. Prentice Hall Europe (1998)
12. ISO/IEC. *Information technology: Z formal specification notation: Syntax, type system and semantics*. ISO/IEC 13568:2002(E) (July 2002)
13. Josephs, M.B.: Functional programming with side-effects. *Science of Computer Programming* 7, 279–296 (1986)
14. Launchbury, J.: Lazy imperative programming. In: Hudak, P. (ed.) *Proceedings of the ACM SIGPLAN Workshop on State in Programming Languages*, Yale University Research Report YALEU/DCS/RR-968, pp. 46–56 (1993)
15. Launchbury, J., Peyton Jones, S.: State in Haskell. *Lisp and Symbolic Computation* 8(4), 293–341 (1995)
16. Nelson, G.: A generalization of Dijkstra’s calculus. *ACM Transactions on Programming Languages and Systems* 11(4), 517–561 (1989)
17. Peyton Jones, S. (ed.): *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press, Cambridge (2003)
18. Plotkin, G.D.: A powerdomain construction. *SIAM Journal on Computing* 5(3), 452–487 (1976)
19. Van Roy, P., Haridi, S.: *Concepts, Techniques, and Models of Computer Programming*. MIT Press, Cambridge (2004)
20. Schmidt, D.A.: *Denotational Semantics: A Methodology for Language Development*. William C. Brown Publishers (1986)
21. Schmidt, G.: Partiality I: Embedding relation algebras. *Journal of Logic and Algebraic Programming* 66(2), 212–238 (2006)
22. Schmidt, G., Hattensperger, C., Winter, M.: Heterogeneous relation algebra. In: Brink, C., Kahl, W., Schmidt, G. (eds.) *Relational Methods in Computer Science*, ch. 3, pp. 39–53. Springer, Wien (1997)
23. Schmidt, G., Ströhlein, T.: *Relationen und Graphen*. Springer, Heidelberg (1989)
24. Schröder, B.S.W.: *Ordered Sets: An Introduction*. Birkhäuser, Basel (2003)
25. Smithson, R.E.: Fixed points of order preserving multifunctions. *Proceedings of the American Mathematical Society* 28(1), 304–310 (1971)

26. Smyth, M.B.: Power domains. *Journal of Computer and System Sciences* 16(1), 23–36 (1978)
27. Søndergaard, H., Sestoft, P.: Non-determinism in functional languages. *The Computer Journal* 35(5), 514–523 (1992)
28. Walker, J.W.: Isotone relations and the fixed point property for posets. *Discrete Mathematics* 48(2–3), 275–288 (1984)
29. Woodcock, J., Davies, J.: *Using Z*. Prentice-Hall, Englewood Cliffs (1996)

The Algebraic Approach I: The Algebraization of the Chomsky Hierarchy

Mark Hopkins

The Federation Archive
federation2005@netzero.net
<http://federation.g3z.com>

Abstract. The algebraic approach to formal language and automata theory is a continuation of the earliest traditions in these fields which had sought to represent languages, translations and other computations as expressions (e.g. regular expressions) in suitably-defined algebras; and grammars, automata and transitions as relational and equational systems over these algebras, that have such expressions as their solutions. The possibility of a comprehensive foundation cast in this form, following such results as the algebraic reformulation of the Parikh Theorem, has been recognized by the Applications of Kleene Algebra (AKA) conference from the time of its inception in 2001.

Here, we take another step in this direction by embodying the Chomsky hierarchy, itself, within an infinite complete lattice of algebras that ranges from dioids to quantales, and includes many of the forms of Kleene algebras that have been considered in the literature. A notable feature of this development is the generalization of the Chomsky hierarchy, including type 1 languages, to arbitrary monoids.

Keywords: Kleene, Language, Context-Free, Regular Expression, Rational, Monoid, Semigroup, Dioid, Quantale, Grammar.

1 The Algebraic Point of View

From its inception, the Applications in Kleene Algebra conference has recognized the possibility of a comprehensive algebraic foundation for formal language and automata theory:

*“Recent algebraic versions of classical results in formal language theory, e.g. Parikh’s theorem [1], point to the exciting possibility of a general algebraic theory that subsumes classical combinatorial automata and formal language theory [pointing] to a much more general, purely axiomatic theory in the spirit of modern algebra.”*¹

An additional step shall be taken in this direction, here, by recasting the the Chomsky hierarchy in algebraic form as an infinite complete lattice of algebras

¹ Programme introduction, Applications of Kleene Algebra, Schloss Dagstuhl, February 2001.

ranging from dioids to quantales. The synthesis provided by the dioid-quantale hierarchy, introduced here, brings fully to bear the power of monads and adjunctions. Much of these developments were foreshadowed by Kozen [5], where implicit use was made of the monad concept to develop a hierarchical relation between different varieties of Kleene algebras. Earlier work has been carried out by Conway [6] in the study of the algebra that came to be known as the Quantale, the $*$ -continuous Kleene Algebra, and the “countably-closed dioid”.

In a separate line of development, the quantale and dioid have also emerged in the 1980’s in Quantum Physics (hence the name “quantale”), particularly in the study of C^* -algebras and von Neumann algebras, in non-linear dynamics, linear logic, Penrose tilings, discrete event systems ([8,9,10,11,12,13,14,15], see also [4] and references contained therein), and related fields (e.g. *Idempotent Analysis* from Maslov [7], et. al.)

1.1 Preliminaries

The notions of semigroups, monoids, partial orderings, semi-lattices and lattices are standard (e.g. [4,16,17]) and will not be dealt with in great depth here.

In the standard formulation of formal languages and automata, which we will refer to henceforth as the classical theory, a language is regarded as a subset of a free monoid $M = X^*$, though more general monoids may sometimes be considered, e.g. Parikh vectors over commutative monoids, translations and relations over direct products of monoids. Different families of *languages* over an alphabet X are then identified as distinguished families of subsets of a monoid X^* .² Along the way, one naturally encounters the issue of closure properties: is a given family closed under substitutions, morphisms, inverse morphisms, products, unions, etc.?

This specificity seems to extend to grammars: curiously, there seems to be an absence of the notion of grammars in the literature other than for free monoids. A formulation suitable for general monoids has therefore been provided in the appendix, where the algebraic concept of free extensions will emerge as a key element.

The monoid product $\cdot : M \times M \rightarrow M$ lifts to a product $\cdot : \mathcal{P}M \times \mathcal{P}M \rightarrow \mathcal{P}M$ over the power set by $AB = \{ab \in M : a \in A, b \in B\}$. This endows the power-set $\mathcal{P}M$ with the structure of a monoid containing that of M in virtue of the correspondence $\eta_M : a \mapsto \{a\}$ which embeds M into $\mathcal{P}M$ by virtue of the relations $\{a\}\{b\} = \{ab\}$ and $\{a\}\{1\} = \{a1\} = \{a\} = \{1a\} = \{1\}\{a\}$. Whereas the product operation may be thought of as embodying the primitive concept of sequentiality, the additional structure provided by the operators $0 = \emptyset$ and $A+B = A \cup B$ may be thought of as giving us a way to embody non-determinism. The ordering relation $A \geq B \Leftrightarrow A \supseteq B$ may then be identified as a precursor of the notions of derivability or transformation $A \rightarrow B$. In this setting, a grammar

² The analogous classification of *translations* from an alphabet X to another alphabet Y is then distinguished by the corresponding families of subsets of the product monoid $X^* \times Y^*$. This generalizes further to relations of ternary or higher degree.

or automaton may then be regarded as a way of writing down a system of relations. The principle of *finite derivability* is encoded by the requirement that the object (language, translation, relation, etc.) represented by the grammar or automaton should be the *least solution* in the corresponding relational system.

This is the essence of what may be termed the *Algebraic Approach*.

However, the definitions in the classical theory are cast almost entirely in set-theoretic terms, as are the arguments for the corresponding theorems, even though the ideas and the results frequently have a purely algebraic flavor and can be stated in such fashion often with both an increase in transparency and generality. As a result, the full potential of the results arrived at classically is missed. This discrepancy is what the algebraic approach seeks to rectify.

1.2 Dioids, Quantales and the Relational View

A *dioid* is also known as an idempotent semiring and may be defined by the identities $a(bc) = (ab)c$, $a1 = a = 1a$, $a + a = a$, $a(b + c)d = abd + acd$, $a + (b + c) = (a + b)c$, $a + 0 = a = 0 + a$, $a + b = b + a$ and $a0b = 0$.

In virtue of idempotency, $a + a = a$, such an algebra may be defined as a partially ordered monoid, with the “natural” partial ordering $a \geq b$ given by $\exists x : a = x + b$, or equivalently by $a = a + b$. Taking the ordering relation as primitive, addition may be defined as the least upper bound, characterizing it by the property that $x \geq a + b$ if and only if $x \geq a$ and $x \geq b$. The minimal element 0 is characterized by the property $x \geq 0$. A consequence of these axioms (see, e.g. [4]) is that both dioid operations $(a, b) \mapsto ab$ and $(a, b) \mapsto a + b$ are monotonic.

The partial ordering enters into formal language theory in various guises as *reducibility*, *derivability*, *transformation*, etc. The addition operator may then be regarded as a representation of the phenomenon of *non-deterministic branching*, the additive identity as that of *failure*.

This view of formal languages as a non-deterministic algebra for words leads to an alternate interpretation of the foregoing. A dioid D is equivalently described as a partially ordered monoid in which every finite subset $A \subseteq D$ has a least upper bound $\sum A \in D$ with $\sum \{a_1, \dots, a_n\} = 0 + a_1 + \dots + a_n$, which is assumed to be finitely distributive with respect to the product. Because of the finite distributivity property, the summation operator $\Sigma : \mathcal{F}D \rightarrow D$ inherited from the semilattice will turn out to be a dioid homomorphism with $\sum (AB) = (\sum A)(\sum B)$, for $A, B \in \mathcal{F}D$; $\sum \{d\} = d$, for $d \in D$; and $a(\sum A)b = \sum (aAb)$, for $A \in \mathcal{F}D$ and $a, b \in D$.

The least upper bound operator $\Sigma : \mathcal{F}D \rightarrow D$ is thus seen to not only be a monoid homomorphism, but the left-adjoint of the monoid embedding $\eta_M : M \rightarrow \mathcal{F}M$ into the family $\mathcal{F}M$ of finite subsets of M .

The existence of such an operator for a given monoid M equivalently identifies M as a dioid. Thus $\mathcal{F}M$, itself, is the free dioid extension of M , and $\mathcal{F}X^*$ is

³ The finite and countable subsets of a given set A will be denoted, respectively, $\mathcal{F}A$ and ωA .

the free dioid generated by a set X . In the context of formal languages, when X is a finite non-empty set representing an alphabet, the family $\mathcal{F}X^*$ may be identified as the family of *finite languages* over the alphabet X . In a more general algebraic context, a family of languages may therefore be regarded as forming a dioid with the additive operator $\cup$, partial ordering relation $\subseteq$, zero element $\emptyset$, multiplicative identity $\{1\}$ and set-wise concatenation as the product.

If least upper bounds exist for arbitrary subsets, with infinite distributivity, the result is the algebra known as a *quantale*.⁴ The free quantale extension of a monoid M is just its powerset $\mathcal{P}M$. Correspondingly, the free quantale $\mathcal{P}X^*$ may be regarded as the general family of languages over X .

A similar consideration applies also to the other dioid varieties corresponding to the operators $M \mapsto \mathcal{R}M$ and $M \mapsto \omega M$ respectively to the rational and countable subsets of M . This leads to corresponding adjunction pairs, respectively, to the $*$ -continuous Kleene algebras and closed semirings.

1.3 Kleene Algebras and Regular Expressions

The “process” view is expanded by treating also the notion of *unbounded repetition* or *iteration* by what is known as the Kleene star operator $a \mapsto a^*$. In the classical interpretation over the power set algebra $\mathcal{P}M$ such an operator may be defined as $A^* = \{1\} \cup \bigcup_{n \geq 0} A^n =$ monoid closure of A . This results in what is known as a *Kleene algebra*, which contains the three operations of the product, sum, star; the injection $\eta_M(M)$ of the underlying monoid M of words; and the distinguished constants $\emptyset, \{1\}$. The Kleene star, A^* is the least upper bound of all the powers A^n as $n = 0, 1, 2, \dots$: $A^* = \sum_{n \geq 0} A^n$. This identity be combined with distributivity to yield what is known as the *$*$ -continuity* property: $\sum_{n \geq 0} AB^n C = AB^* C$.

For a given monoid, M , the closure of the family $\mathcal{F}M$ under products, finite unions and the Kleene star yield what are known as the *rational subsets* of M , which we will denote $\mathcal{R}M$. In particular, the families $\mathcal{R}X^*$ and $\mathcal{R}(X^* \times Y^*)$, for finite non-empty alphabets X and Y give us, respectively, the regular languages over X and rational transductions from X to Y .

There are many possible and inequivalent ways to formulate a theory of regular expressions which each have the property of capturing all the identities which hold in the standard set-theoretic interpretation. Two early examples were developed in [2,3].

As shown in [2], $*$ -continuous Kleene algebras are equivalently defined as partially ordered monoids in which the least upper bound property and distributivity property apply to the rational subsets. The corresponding homomorphisms are described equivalently as maps that preserve the Kleene operators, or as monoid homomorphisms that preserve least upper bounds for rational subsets.

⁴ Generally, one distinguishes between quantales with or without the multiplicative unit 1. Our focus, here, shall be exclusively on the latter variety, the unital quantales, which we shall, for brevity, refer to as just quantales.

Through a standard construction, by ideals, a $*$ -continuous Kleene algebra may be extended to one which possesses similar properties for its countable subsets (the closed semiring). From here, in turn, a further extension may be formulated, leading to a quantale structure.

2 The Dioid-Quantale Hierarchy

Inevitably, this leads to the question: what other types of “subset families” can we define and incorporate into this hierarchy?

2.1 Monadic Operators

We start by defining a monadic dioid D as a dioid in which the formal sum exists for all members of a distinguished family of subsets $\mathcal{A}D$, with respect to which distributivity also holds. In order to arrive at a consistent formulation, particularly one that admits a construction of adjunctions, we will need to place restrictions on the operator $\mathcal{A}$, as follows:

Definition 1. *A monadic operator $\mathcal{A}$ is a monoid operator satisfying the properties*

- $\mathbf{A}_0$ $\mathcal{A}M$ is a family of subsets of the monoid M
- $\mathbf{A}_1$ $\mathcal{A}M$ contains all the finite subsets of M
- $\mathbf{A}_2$ $\mathcal{A}M$ is closed under products, thus making $\mathcal{A}M$ a monoid.
- $\mathbf{A}_3$ $\mathcal{A}M$ is closed under unions of subsets from $\mathcal{A}M$
- $\mathbf{A}_4$ $\mathcal{A}$ respects monoid homomorphisms. If $f : M \rightarrow N$ is a monoid homomorphism, then $\mathcal{A}\tilde{f}(U) \in \mathcal{A}N$ for all $U \in \mathcal{A}M$.

For any monoid operator $\mathcal{A}$, the following may then be defined:

Definition 2. *Let M be a partially ordered monoid and write $x > A$, if x is an upper bound of a set A . Then*

- $\mathbf{D}_0$ M is $\mathcal{A}$ -additive if every $U \in \mathcal{A}M$ has a least upper bound $\sum U \in M$,
- $\mathbf{D}_1$ M is $\mathcal{A}$ -separable if for all $x > aUb$ there exists $u > U$ such that $x \geq aub$, where $a, b \in M$ and $U \in \mathcal{A}M$.
- $\mathbf{D}_2$ M is strongly $\mathcal{A}$ -separable if for all $x > UV$ there exist $u > U, v > V$ such that $x \geq uv$, where $U, V \in \mathcal{A}M$.
- $\mathbf{D}_3$ A monoid homomorphism $f : M \rightarrow N$ is $\mathcal{A}$ -additive if for all $y > \tilde{f}(U)$ there exists $x > U$ such that $y \geq f(x)$, where $U \in \mathcal{A}M$.

One may verify that when the monoid is $\mathcal{A}$ -additive then both forms of separability become equivalent to each other and to the following conditions

- $\mathbf{D}'_1$ $a, b \in M \& U \in \mathcal{A}M \rightarrow \sum (aUd) = a(\sum U)d$ (strong distributivity),
- $\mathbf{D}'_2$ $U, V \in \mathcal{A}M \rightarrow \sum (UV) = \sum U \cdot \sum V$ (distributivity).

⁵ In here, and in the following, we will denote the image of a function f on a set U by $\tilde{f}(U) \equiv \{f(u) : u \in U\}$.

For order-preserving monoid homomorphisms, $f : M \rightarrow M'$, $\mathcal{A}$ -additivity reduces equivalently to the condition

$$\mathbf{D}'_3 \quad U \in \mathcal{A}M \rightarrow f(\sum U) = \sum \tilde{f}(U).$$

Therefore, we are led to the following definitions:

Definition 3. An $\mathcal{A}$ -diod is a partially ordered monoid M satisfying $\mathbf{D}_0$ and $\mathbf{D}_1$ (or any of its equivalents, $\mathbf{D}_2$, $\mathbf{D}'_1$, $\mathbf{D}'_2$) with respect to $\mathcal{A}$; i.e., a dioid that is both $\mathcal{A}$ -additive and $\mathcal{A}$ -separable. An $\mathcal{A}$ -morphism is an order-preserving monoid homomorphism $f : M \rightarrow N$ that satisfies $\mathbf{D}_3$ (or equivalently, $\mathbf{D}'_3$).

The following results may then be formulated:

Theorem 1. $\mathcal{A}M$ is an $\mathcal{A}$ -diod for any monoid M .

Proof. The least upper bound operator in $\mathcal{A}M$ is just set union. Property $\mathbf{A}_3$ guarantees that every member of $\mathcal{A}M$ has a union in $\mathcal{A}M$, thus satisfying $\mathbf{D}_0$. Property $\mathbf{D}'_2$ reduces to the requirement that $\bigcup UV = \bigcup U \cdot \bigcup V$, which is verified by the following chain of equivalences

$$x \in \bigcup U \cdot \bigcup V \leftrightarrow \exists A \in U, B \in V : x \in AB \leftrightarrow \exists C \in UV : x \in C \leftrightarrow x \in \bigcup UV.$$

Theorem 2. $\Sigma : \mathcal{A}D \rightarrow D$ is an $\mathcal{A}$ -morphism for any $\mathcal{A}$ -diod D .

Proof. Suppose that D is an $\mathcal{A}$ -diod. Then we immediately see that $\Sigma : \mathcal{A}D \rightarrow D$ is a monoid homomorphism. Property $\mathbf{D}'_3$ then reduces to the requirement that $\sum \bigcup Y = \sum_{V \in Y} \sum V$ for $Y \in \mathcal{A}AD$ i.e.,

$$\sup \bigcup Y = \sup \{ \sup V : V \in Y \},$$

which is a general property of partially ordered sets. We note that we only need to stipulate the existence of one side of the equation, and of $\sup V$, for each $V \in Y$. Then both sides will be well-defined.

Theorem 3. Every monoid homomorphism $f : M \rightarrow N$ lifts to an $\mathcal{A}$ -morphism $\tilde{f} : \mathcal{A}M \rightarrow \mathcal{A}N$.

Proof. Let $f : M \rightarrow N$ be a monoid homomorphism. Then $\tilde{f} : \mathcal{A}M \rightarrow \mathcal{A}N$ is also one since $\tilde{f}(\{1\}) = \{f(1)\} = \{1\}$ and

$$\tilde{f}(UV) = \{f(uv) : u \in U, v \in V\} = \{f(u)f(v) : u \in U, v \in V\} = \tilde{f}(U)\tilde{f}(V).$$

The requirement that least upper bounds from $\mathcal{A}M$ also be preserved is given by $\mathbf{D}'_3$, which takes on the following form here $\tilde{f}(\bigcup Y) = \bigcup \tilde{f}(Y)$ for $Y \in \mathcal{A}AM$.

This is also a general property of sets.

Theorem 4 (The Universal Property). The free $\mathcal{A}$ -diod extension of a monoid M is $\mathcal{A}M$.

Equivalently, this may be stated as: (a) $\eta_M : M \rightarrow \mathcal{A}M, m \mapsto \{m\}$ is a monoid homomorphism, and (b) a monoid homomorphism $f : M \rightarrow D$ to an $\mathcal{A}$ -dioid D extends uniquely to an $\mathcal{A}$ -morphism $f^* : \mathcal{A}M \rightarrow D$; i.e., such that $f = f^* \circ \eta_M$.

Proof. This is an immediate consequence of theorems 2 and 3 with the homomorphism given by $f^* = \sum \tilde{f}$. Uniqueness is proven as follows. The equality $f^* = \sum \tilde{f}$ is already established on finite subsets for any morphism f^* satisfying the property $f = f^* \circ \eta_M$. To show that $f^*(U) = \sum \tilde{f}(U)$ for a (possibly infinite) $U \in \mathcal{A}M$, consider first the image $\hat{U} = \widetilde{\eta_M}(U) \in \mathcal{A}AM$. This is a family of singleton subsets. Therefore,

$$\tilde{f}^*(\hat{U}) = \left(\sum \tilde{f} \right) (\hat{U}) .$$

By $\mathcal{A}$ -continuity of f^* , it then follows that

$$f^*(U) = f^*\left(\bigcup \hat{U}\right) = \sum \tilde{f}^*(\hat{U}) = \sum \left(\sum \tilde{f} \right) (\hat{U})$$

while

$$\sum \tilde{f}(U) = \sum \tilde{f}\left(\bigcup \hat{U}\right) = \sum \left(\sum \tilde{f} \right) (\hat{U}) .$$

Example 1. A hierarchy of $\mathcal{A}$ -dioids is provided in the following table

$\mathcal{A}$ Description	Structure
$\mathcal{F}$ Finite subsets	Dioid
$\mathcal{R}$ Rational subsets	*-Continuous Kleene Algebra
$\mathcal{C}$ Context-free subsets	“Algebraic Dioid”
$\mathcal{S}$ Context-sensitive subsets	“Context-Sensitive Dioid”
$\mathcal{T}$ Turing-computable subsets	“Transcendental Dioid”
ω Countable subsets	Closed semiring
$\mathcal{P}$ Power set	Quantale (with unit)

More generally, one may readily verify that monadic operators preserve submonoid ordering (in virtue of $\mathbf{A}_4$) and are closed under arbitrary intersections. Therefore, we have the following results.

Theorem 5. *Monadic operators respect submonoid ordering: if $M \subseteq M'$, then $\mathcal{A}M \subseteq \mathcal{A}M'$.*

Proof. This is the direct result of applying $\mathbf{A}_4$ to the inclusion homomorphism $i : m \in M \mapsto m \in M'$.

Theorem 6 (Hierarchical Completeness). *Monadic operators form a complete lattice with top $\mathcal{A}M = \mathcal{P}M$ and bottom $\mathcal{A}M = \mathcal{F}M$.*

Proof. Let $\mathbf{Z}$ be a family of monadic operators, and define $(\wedge \mathbf{Z}) M = \bigcap_{\mathcal{A} \in \mathbf{Z}} \mathcal{A}M$. In the special case $\mathbf{Z} = \emptyset$, we define $\wedge \mathbf{Z} = \mathcal{P}$, which trivially satisfies the defining properties for a monadic operator. Otherwise, suppose $\mathbf{Z} \neq \emptyset$. Properties $\mathbf{A}_0$,

$\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_4$ are then easily verified for $\wedge Z$. Property $\mathbf{A}_3$, however, is not as immediate. For, suppose that $\mathcal{A} \in Z$, we then have

$$(\wedge Z) M = \bigcap_{\mathcal{A} \in Z} \mathcal{A} M \subseteq \mathcal{A} M.$$

To complete the proof, we need to make use of the preservation of submonoid ordering under monadic operators (theorem 5). Then, we may write $\mathcal{A}(\wedge Z) M \subseteq \mathcal{A} \mathcal{A} M$. Thus, for any family of subsets $Y \in (\wedge Z)(\wedge Z) M$, we have that

$$Y \in \bigcap_{\mathcal{A} \in Z} \mathcal{A}(\wedge Z) M \subseteq \mathcal{A}(\wedge Z) M \subseteq \mathcal{A} \mathcal{A} M.$$

Thus, by $\mathbf{A}_3$, $\bigcup Y \in \mathcal{A} M$. Since $\mathcal{A} \in Z$ was arbitrarily chosen, this shows that

$$\bigcup Y \in \bigcap_{\mathcal{A} \in Z} \mathcal{A} M = (\wedge Z) M.$$

Thus $\wedge Z$ satisfies property $\mathbf{A}_3$.

2.2 Closure Under Substitutions

Examples 1 suggest that monadic operators provide us with an algebraic generalization of the classical concept of language family. Properties $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_4$ are well known in the classical setting and are readily established for each of the examples. However, $\mathbf{A}_3$ is decidedly non-classical, and cannot even be expressed in that setting, though it may also be established for ωM by a well-known classical proof, and analogously for $\mathcal{T}M$. The cases $\mathcal{R}M$, $\mathcal{C}M$ and $\mathcal{S}M$ require further elaboration.

In fact, there is a classical analogue closely related to $\mathbf{A}_3$ that also happens to subsume $\mathbf{A}_4$. This relates to the concept of a *substitution*. Given two monoids M, N , a substitution map $\sigma : M \rightarrow \mathcal{P}N$ is thought of as a map which replaces each element of M by a *language* in N . Reflecting the hierarchy of $\mathcal{A}$ -dioids is a hierarchy of substitutions, determined by the range of the map. This leads to the following definition.

Definition 4. *Let M, N be monoids. A monoid homomorphism $\sigma : M \rightarrow \mathcal{P}N$ is called a substitution. In addition, if $\mathcal{A}N \subseteq \mathcal{P}N$ is any family of subsets such that $\sigma(m) \in \mathcal{A}N$, for each $m \in M$, then σ will be called an $\mathcal{A}$ -substitution.*

Every substitution $\sigma : M \rightarrow \mathcal{P}N$ leads uniquely to a map between the respective power sets of the monoids given, for $A \in \mathcal{P}M$, by $\hat{\sigma}(A) = \bigcup_{m \in A} \sigma(m) \in \mathcal{P}N$. Moreover, it follows directly from this definition that this map distributes over unions: $\hat{\sigma}(\bigcup Y) = \bigcup_{A \in Y} \hat{\sigma}(A)$, for all $Y \subseteq \mathcal{P}M$. Therefore it is a quantale homomorphism between the respective power sets. This leads to the following result.

Theorem 7. *A substitution $\sigma : M \rightarrow \mathcal{P}N$ determines and is uniquely determined by a quantale homomorphism $\phi : \mathcal{P}M \rightarrow \mathcal{P}N$ such that $\phi(\{m\}) = \sigma(m)$ for $m \in M$.*

Proof. In the forward direction, we have

$$\hat{\sigma} \left(\bigcup Y \right) = \bigcup_{m \in \bigcup Y} \sigma(m) = \bigcup_{A \in Y} \bigcup_{m \in A} \sigma(m) = \bigcup_{A \in Y} \hat{\sigma}(A),$$

for $Y \subseteq \mathcal{P}M$, and $\hat{\sigma}(\{m\}) = \bigcup_{m' \in \{m\}} \sigma(m') = \sigma(m)$, for $m \in M$. Conversely, suppose $\phi : \mathcal{P}M \rightarrow \mathcal{P}N$ is a quantale homomorphism satisfying the stated condition. Then for $A \subseteq M$, making use of the invariance property, we have $\phi(A) = \bigcup_{m \in A} \phi(\{m\}) = \bigcup_{m \in A} \sigma(m) = \hat{\sigma}(A)$.

With these preliminaries, we may then state the following alternative to properties **A**₃ and **A**₄

A₅ $\mathcal{A}$ respects $\mathcal{A}$ -substitutions. If $\sigma : M \rightarrow \mathcal{P}N$ is an $\mathcal{A}$ -substitution, then $\hat{\sigma}(U) \in \mathcal{A}N$ for all $U \in \mathcal{A}M$.

We may then establish the equivalence between the two sets of properties as follows:

Theorem 8. *Let $\mathcal{A}$ be a monoid operator satisfying **A**₀, **A**₁ and **A**₂. Then **A**₃ and **A**₄ are equivalent to **A**₅.*

Proof. In the following, let M, N be monoids.

First, we will establish **A**₀, **A**₂, **A**₃, **A**₄ $\rightarrow$ **A**₅. Suppose $\sigma : M \rightarrow \mathcal{P}N$ is an $\mathcal{A}$ -substitution and $U \in \mathcal{A}M$. Then, by **A**₀, **A**₂, $\mathcal{A}N \subseteq \mathcal{P}N$ is a monoid with $\sigma : M \rightarrow \mathcal{A}N$ a monoid homomorphism. By property **A**₄, it follows that $\tilde{\sigma}(U) = \{\sigma(m) : m \in U\} \in \mathcal{A}N$. In turn, by **A**₃, it follows that $\hat{\sigma}(U) = \bigcup \tilde{\sigma}(U) \in \mathcal{A}N$.

Second, we will prove that **A**₁, **A**₅ $\rightarrow$ **A**₄. Suppose $f : M \rightarrow N$ is a monoid homomorphism and $U \in \mathcal{A}M$. Then by **A**₁, $\mathcal{A} \geq \mathcal{F}$, therefore $\sigma : m \in M \mapsto \{f(m)\} \in \mathcal{F}N \subseteq \mathcal{A}N$ is an $\mathcal{A}$ -substitution. Applying **A**₅, it follows that $\hat{\sigma}(U) \in \mathcal{A}N$. But

$$\hat{\sigma}(U) = \bigcup_{m \in U} \sigma(m) = \bigcup_{m \in U} \{f(m)\} = \{f(m) : m \in U\} = \tilde{f}(U).$$

Thus $\tilde{f}(U) \in \mathcal{A}N$.

Finally, we will show that **A**₀, **A**₂, **A**₅ $\rightarrow$ **A**₃. Suppose $Y \in \mathcal{A}AM$. Then, by **A**₂, the identity map $\sigma = I_{\mathcal{A}M} : \mathcal{A}M \rightarrow \mathcal{A}M$ is a monoid homomorphism; and, by **A**₀, $\sigma : \mathcal{A}M \rightarrow \mathcal{A}M \subseteq \mathcal{P}M$ is an $\mathcal{A}$ -substitution. Therefore, applying **A**₅, it follows that $\hat{\sigma}(Y) \in \mathcal{A}M$. But

$$\hat{\sigma}(Y) = \bigcup_{U \in Y} \sigma(U) = \bigcup_{U \in Y} U = \bigcup Y.$$

Therefore, $\bigcup Y \in \mathcal{A}M$.

2.3 Closure under Inverse Morphisms

An important application of the universal property (theorem 4) is the following:

Theorem 9 ($\mathbf{A}_6$). *If the monoid homomorphism $f : M \rightarrow N$ is surjective, then so is the lift $\tilde{f} : \mathcal{A}M \rightarrow \mathcal{A}N$.*

Proof. Assume the conditions stated hold. The property of surjectivity may be stated solely in terms of the properties of homomorphisms in the following way: given homomorphisms $g, h : N \rightarrow P$ to another monoid P , if $g \circ f = h \circ f$ then $g = h$. Surjectivity for the lifting is proven via the analogous property. Assume that $g, h : \mathcal{A}N \rightarrow D$ are now $\mathcal{A}$ -morphisms to an $\mathcal{A}$ -diod D , such that $g \circ \tilde{f} = h \circ \tilde{f}$. Then, we may push this back to a map on the monoid M and write $g \circ f \circ \eta_M = h \circ f \circ \eta_M$. But, $f \circ \eta_M = \eta_M \circ f$, therefore $g \circ \eta_M \circ f = h \circ \eta_M \circ f$. From the surjectivity of f , it follows that $g \circ \eta_M = h \circ \eta_M$. The universal property, theorem 4, asserts that the extension of this map $g \circ \eta_M = h \circ \eta_M : N \rightarrow D$ to a map on $\mathcal{A}N$ is unique. Therefore, $g = h$. Thus, $\tilde{f}$ is surjective.

As a consequence, we find that monadic operators respect inverse morphisms in the following sense:

Theorem 10. *Let $\mathcal{A}$ be a monadic operator. Then if $f : M \rightarrow N$ is a surjective monoid homomorphism, and $V \in \mathcal{A}N$ then $V = f(U)$ for some $U \in \mathcal{A}M$. Moreover, there is a monoid $\hat{N}$, a surjective map $\sigma : \hat{N} \rightarrow N$, and a factoring $\sigma = f \circ \phi$ into $\phi : \hat{N} \rightarrow M$ and f ; such that each $V \in \mathcal{A}N$ may be expressed as $\sigma(\hat{V})$ for some $\hat{V} \in \mathcal{A}\hat{N}$ where $\phi(\hat{V}) \in \mathcal{A}M$.*

Proof. The first statement is a direct consequence of our previous result, theorem 9. For the second part, let $Y \subseteq N$ be any generating subset of the monoid N . The universal property for free monoids then associates a canonical monoid homomorphism $\sigma : \hat{N} = Y^* \rightarrow N$ with the inclusion $\sigma : Y \rightarrow N$. This maps the free monoid Y^* generated by the set Y onto the closure of that set within N , which (by assumption) is just N , itself.

In its greatest generality, this argument requires the Axiom of Choice. If Y is an infinite set, then for each $y \in Y$, we need to choose an element $m \in M$ such that $f(m) = \sigma(y)$, and then define $\phi(y) = m$. However, for the operators $\mathcal{A} = \mathcal{F}, \mathcal{R}, \mathcal{C}, \mathcal{S}, \mathcal{T}$, we will always be able to express a subset $V \in \mathcal{A}N$ as $V \in \mathcal{A}Y^*$ for some finite subset $Y \subseteq N$. However, this property ($\mathbf{A}_7$: *finite generativity*) will not have any bearing on our results, so we will not further elaborate on it here.

Let $V \in \mathcal{A}N$. Since $\sigma : \hat{N} \rightarrow N$ is surjective then there exists $\hat{V} \in \mathcal{A}\hat{N}$ such that $V = \sigma(\hat{V})$. The remainder of the theorem then follows by property $\mathbf{A}_4$.

3 Inclusion of the Chomsky Hierarchy

Up to now, we have left the issue unresolved of which of our examples actually constitute monadic operators. Properties $\mathbf{A}_0$ and $\mathbf{A}_1$ are true, by construction,

for each operator in example [11](#). Similarly, $\mathbf{A}_2$, $\mathbf{A}_3$ and $\mathbf{A}_4$ are well-known and easily proven in the cases of $\mathcal{F}$, ω and $\mathcal{P}$.

However, for $\mathcal{R}$, $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{T}$, $\mathbf{A}_3$ is neither obvious nor well-known, while $\mathbf{A}_2$ and $\mathbf{A}_4$ require further clarification. This is what we will resolve here.

3.1 The $\mathcal{R}$ Operator and *-Continuous Kleene Algebras

A *-continuous Kleene algebra is a dioid which is $\mathcal{R}$ -additive. By definition, the rational subsets $\mathcal{R}M$ of a monoid M are the closure of $\mathcal{F}M$ under the product, finite union and Kleene star. Therefore $\mathbf{A}_2$ is satisfied, so that we need only prove $\mathbf{A}_3$ and $\mathbf{A}_4$, or equivalently $\mathbf{A}_5$.

Theorem 11. *The operator $\mathcal{R}$ satisfies $\mathbf{A}_5$.*

Proof. This may be shown by induction. Let $\sigma : M \rightarrow \mathcal{R}N$ be an $\mathcal{R}$ -substitution from a monoid M to the rational subsets of a monoid N . For finite sets $U \in \mathcal{F}M$, we immediately have $\hat{\sigma}(U) = \bigcup_{u \in U} \sigma(u) \in \mathcal{R}N$, since $\mathcal{R}N$ is closed under finite unions. Moreover, we may show that $\hat{\sigma}$ preserves unions and products, since $\hat{\sigma}(\bigcup Y) = \bigcup_{u \in \bigcup Y} \sigma(u) = \bigcup_{U \in Y} \hat{\sigma}(U)$, for any $Y \in \mathcal{P}M$, while for $U, V \subseteq M$,

$$\hat{\sigma}(UV) = \bigcup_{u \in U, v \in V} \sigma(uv) = \bigcup_{u \in U} \sigma(u) \bigcup_{v \in V} \sigma(v) = \hat{\sigma}(U) \hat{\sigma}(V) .$$

From this, it follows that $\hat{\sigma}$ preserve Kleene stars,

$$\hat{\sigma}(U^*) = \hat{\sigma}\left(\bigcup_{n \geq 0} U^n\right) = \bigcup_{n \geq 0} \hat{\sigma}(U^n) = \bigcup_{n \geq 0} \hat{\sigma}(U)^n = \hat{\sigma}(U)^* .$$

Consequently, if we let $U, V \in \mathcal{R}M$ and assume by inductive hypothesis that $\hat{\sigma}(U) \in \mathcal{R}N$ and $\hat{\sigma}(V) \in \mathcal{R}N$, then it follows that $\hat{\sigma}(UV) = \hat{\sigma}(U) \hat{\sigma}(V) \in \mathcal{R}N$, $\hat{\sigma}(U \cup V) = \hat{\sigma}(U) \cup \hat{\sigma}(V) \in \mathcal{R}N$ and $\hat{\sigma}(U^*) = \hat{\sigma}(U)^* \in \mathcal{R}N$, since $\mathcal{R}N$ is closed under products, finite unions and Kleene stars.

3.2 The $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{T}$ Operators

There remains the issue of $\mathbf{A}_2$, $\mathbf{A}_3$ and $\mathbf{A}_4$ with respect to $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{T}$.

For the properties $\mathbf{A}_3$ and $\mathbf{A}_4$, it turns out, again, to be more useful to establish $\mathbf{A}_5$, instead. We do this explicitly here for the operator $\mathcal{C}$, closely following the development of the analogous result in the classical theory (c.f. [\[20\]](#) theorem 9.2.2).

Lemma 1 (*The Composition Lemma*). *Let M be a monoid $G = (Q, S, H)$ be a context-free grammar over X^* for a finite $X \subseteq M$. Let $\sigma : M \rightarrow \mathcal{P}N$ be a context-free substitution to the monoid N . For each $x \in X$, let $G_x = (Q_x, S_x, H_x)$ be a context-free grammar such that $L(G_x) = \sigma(x)$; with the sets Q and Q_x for each $x \in X$ all mutually disjoint.*

Define the composition of the grammars⁶ by

$$G \circ_{\sigma} \bigcup_{x \in X} G_x = \left(Q \cup \bigcup_{x \in X} Q_x, \bar{\sigma}(S), \{(q, \bar{\sigma}(\beta)) : (q, \beta) \in H\} \cup \bigcup_{x \in X} H_x \right),$$

where $\bar{\sigma} : X^*[Q] \rightarrow N[Q \cup \bigcup_{x \in X} Q_x]$ is the monoid homomorphism given by $\bar{\sigma}(x) = S_x$ for $x \in X$ and $\bar{\sigma}(q) = q$ for $q \in Q$. Then $L(G \circ \bigcup_{x \in X} G_x) = \sigma(L(G))$.

Proof: Let G' denote the composition. It is an easy induction to show, for each $x \in X$, that $\alpha \rightarrow \beta$ in G_x if and only if $\alpha \rightarrow \beta$ in G' , where $\alpha, \beta \in N[Q_x]$. This makes use of the mutual disjointness of the sets Q_x . The only rules that can apply here are therefore those from H_x . From this, it follows that $[S_x]_{G'} = [S_x]_{G_x} = L(G_x) = \sigma(x)$, for $x \in X$. In a similar way, one may readily verify that $\alpha \rightarrow \beta$ in G if and only if $\bar{\sigma}(\alpha) \rightarrow \bar{\sigma}(\beta)$ in G' . Again, making use of the disjointness of the set Q from all the other sets Q_x , it follows that $[q]_{G'} = \bigcup_{w \in [q]_G} [\bar{\sigma}(w)]_{G'}$, since occurrences of variables of Q in a configuration α must be handled by the rules from H .

From $[\bar{\sigma}(x)]_{G'} = [S_x]_{G'} = \sigma(x)$ ($x \in X$), it follows by inductive argument⁷ that $[\bar{\sigma}(w)]_{G'} = \sigma(w)$, for $w \in X^*$. Using this result, we then have

$$[\bar{\sigma}(q)]_{G'} = [q]_{G'} = \bigcup_{w \in [q]_G} [\bar{\sigma}(w)]_{G'} = \bigcup_{w \in [q]_G} \sigma(w) = \hat{\sigma}([q]_G),$$

for all $q \in Q$. Thus, we have

$$L(G') = [S]_{G'} = [\bar{\sigma}(S)]_{G'} = \hat{\sigma}([S]_{G'}) = \hat{\sigma}(L(G)).$$

To fully establish our results, we need to ensure that (i) such mutually disjoint sets can be chosen, as required by the lemma; and (ii) that a (finite) context-free grammar can be presented as a grammar over a finitely generated submonoid. Property (ii) is a consequence of the fact that only a finite subset $X \subseteq M$ will appear on the right-hand side of the rules of H in a grammar $G = (Q, S, H)$ over the monoid M , since H is finite. Property (i) makes use of the following technical lemma.

Lemma 2 (Substitution Invariance). *Let $G = (Q, S, H)$ be an arbitrary grammar over a monoid M , $\sigma : Q \rightarrow R$ a bijection and*

$$G_{\sigma} = (R, \sigma(S), \{(\sigma(a), \sigma(b)) : (a, b) \in H\}),$$

where $\sigma : M[Q] \rightarrow M[R]$ is the extension to a monoid homomorphism given by $\sigma(m) = m$ for $m \in M$. Then $\alpha \rightarrow \beta$ in G iff $\sigma(\alpha) \rightarrow \sigma(\beta)$ in G_{σ} , for all $\alpha, \beta \in M[Q]$. Moreover, $[\alpha]_G = [\sigma(\alpha)]_{G_{\sigma}}$ for all $\alpha \in M[Q]$. In particular, $L(G) = L(G_{\sigma})$.

⁶ This is the grammar obtained by replacing each terminal x of the grammar G by the start symbol S_x of grammar G_x and combining the non-terminals of G with those of each G_x .

⁷ This makes use of the property $[\alpha\beta]_{G'} = [\alpha]_{G'}[\beta]_{G'}$ which may be proven by induction for context-free grammars G' .

Proof. Since the map σ is a bijection, we only need to show that if $\alpha \rightarrow \beta$ in G , then $\sigma(\alpha) \rightarrow \sigma(\beta)$ in G_σ . This is an easy induction over the structure of derivations.

The converse property follows by considering the inverse σ^{-1} . The remaining statements are then a direct consequence since, for $m \in M$, we have $m \in [\alpha]_G$ iff $\alpha \rightarrow m$ in G iff $\sigma(\alpha) \rightarrow \sigma(m) = m$ in G_σ iff $m \in [\sigma(\alpha)]_{G_\sigma}$. From this, it will then follow that $L(G) = [S]_G = [\sigma(S)]_{G_\sigma} = L(G_\sigma)$.

The proof of **A₂** closely follows that of the classical result. Given subsets $L(G_1)$, $L(G_2)$ of M generated by context-free grammars $G_i = (Q_i, S_i, H_i)$ over M ($i = 1, 2$), one constructs a grammar $G = (Q, S, H)$ for the product by taking $Q = Q_1 \cup Q_2 \cup \{S\}$, $H = H_1 \cup H_2 \cup \{(S, S_1 S_2)\}$, choosing S such that $S \notin Q_1 \cup Q_2 \cup M$. We may then use the property $[\alpha\beta] = [\alpha][\beta]$ to show that $L(G) = [S_1 S_2]_G = [S_1]_{G_1} [S_2]_{G_2} = L(G_1) L(G_2)$.

With these preliminaries established, we then have the following corollary.

Corollary 1. *The operator $\mathcal{C}$ is monadic.*

Though the Composition Lemma and product construction are formulated explicitly for $\mathcal{C}$, it can be refined to make it applicable to $\mathcal{S}$ and $\mathcal{T}$. We'll explain how this may be done for the Composition Lemma. A similar consideration holds for the product construction.

To avoid the need for the property $[\alpha\beta] = [\alpha][\beta]$, the grammar G_x over the monoid N is modified to a grammar over a copy N_x of N . Without loss of generality, we may assume that N is generated by a finite set $Y \subseteq N$, similarly N_x by $Y_x \subseteq N_x$. We must then add rules $n_x \rightarrow n$ to map the copy $n_x \in N_x$ of $n \in N$ to n .

For $\mathcal{S}$, in the Composition Lemma, we will also need to prove the context-sensitivity of grammar G' . First, the set X , will be atomic with respect to a given measure over the monoid M . We may also assume that the elements of X are of unit norm or greater, by rescaling the norm. For each $x \in X$, the starting configuration S_x for each grammar G_x will have a norm of at least 1, thereby ensuring the context-sensitivity of the composition of the grammars. In particular, we will have $\|\alpha\| \leq \|\bar{\sigma}(\alpha)\|$, with respect to suitably defined norms. This leads to the following results:

Corollary 2. *The operators $\mathcal{S}$ and $\mathcal{T}$ are monadic.*

4 Concluding Remarks

The Chomsky Hierarchy is the foundation of both the theory of computation and linguistics. What we have shown is that the hierarchy may be encapsulated and generalized in algebraic form as a hierarchy of algebras. At the bottom of the hierarchy is the dioid, or idempotent semiring. Associated with this is the functor $\mathcal{F}$, which maps a given monoid M to its dioid of finite subsets. Thus, the dioid may be regarded as an algebraization of the concept of finite language. At the

top of the hierarchy is the unital quantale, which is associated with the functor $\mathcal{P}$ that maps a monoid M to its quantale of subsets. Here, the corresponding classical concept is the general language. In between these two extremes are other algebras, corresponding to other monadic operators, which include operators that generalize the 4 levels of the Chomsky hierarchy: $\mathcal{R} < \mathcal{C} < \mathcal{S} < \mathcal{T}$.

In the sequel paper, we show that this hierarchy is complemented by a hierarchy of adjunctions with the properties that

- if $\mathcal{A} \leq \mathcal{B}$ then there exists an adjunction $(\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}, \mathbf{Q}_{\mathcal{B}}^{\mathcal{A}})$.
- if $\mathcal{A} \leq \mathcal{B} \leq \mathcal{C}$ then $\mathbf{Q}_{\mathcal{B}}^{\mathcal{C}} \circ \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} = \mathbf{Q}_{\mathcal{A}}^{\mathcal{C}}$ and $\mathbf{Q}_{\mathcal{B}}^{\mathcal{A}} \circ \mathbf{Q}_{\mathcal{C}}^{\mathcal{B}} = \mathbf{Q}_{\mathcal{C}}^{\mathcal{A}}$.

The functor $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}$ extends each $\mathcal{A}$ -dioid to its $\mathcal{B}$ -completion, and is complemented by the forgetful functor $\mathbf{Q}_{\mathcal{B}}^{\mathcal{A}}$, which maps a $\mathcal{B}$ -dioid D to itself, where the least upper bound operator Σ is restricted to the family $\mathcal{AD}$.

Finally, a few additional comments are in order regarding the algebraic representation of context sensitivity. The unusual way in which ϵ -rules enter into the formulation of context-sensitivity indicates that a more natural setting may be found within semigroup theory. This suggests a parallel formulation of monadic semigroup operators with analogous properties $\mathbf{A}_0$ – $\mathbf{A}_4$ stated for semigroups. One should then be able to prove that if $\mathcal{A}$ is a monadic semigroup operator, then its ϵ -extension

$$\mathcal{A}_{\epsilon}M \equiv \mathcal{A}M \cup \{U \cup \{1\} : U \in \mathcal{A}M\}$$

is a monadic monoid operator; particularly, that it satisfies properties $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_5$.

Finally, in classical theory an equivalence between context-sensitive grammars and non-erasing grammars can be proven [18]. Our definition of context-sensitivity is with respect to non-erasing grammars. One needs to separately prove their equivalence within the broader setting provided here. A similar observation holds concerning the need to verify that the normal forms and conversions of the classical theory (e.g., Chomsky and Greibach normal forms, Kuroda normal form) will continue to hold for generalized grammars.

References

1. Hopkins, M.W., Kozen, D.: Parikh's Theorem in Commutative Kleene Algebra. In: LICS 1999, pp. 394–401 (1999)
2. Kozen, D.: The Design and Analysis of Algorithms. Springer, Heidelberg (1992)
3. Kozen, D.: A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. Information and Computation 110, 366–390 (1994)
4. Gunawardena, J. (ed.): Idempotency. Publications of the Newton Institute. Cambridge University Press, Cambridge (1998)
5. Kozen, D.: On Kleene Algebras and Closed Semirings. In: Rovan, B. (ed.) MFCS 1990. LNCS, vol. 452, pp. 26–47. Springer, Heidelberg (1990)
6. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall, London (1971)

7. Maslov, V.P., Samborskii, S.N. (eds.): Advances in Soviet Mathematics, vol. 13 (1992)
8. Abramsky, S., Vickers, S.: Quantales. observational logic and process semantics Mathematical Structures in Computer Science 3, 161–227 (1993)
9. Vickers, S.: Topology via Logic. In: Cambridge Tracts in Theoretical Computer Science, vol. 5, Cambridge University Press, Cambridge (1989)
10. Mulvey, C.J.: Quantales. Springer Encyclopaedia of Mathematics (2001)
11. Baccelli, F., Mairesse, J.: Ergodic theorems for stochastic operators and discrete event systems. In: [4]
12. Golan, J.S.: Semirings and their applications. Kluwer Academic Publishers, Dordrecht (1999)
13. Yetter, D.N.: Quantales and (Noncommutative) Linear Logic. J. of Symbolic Logic 55, 41–64 (1990)
14. Hoefft, H.: A normal form for some semigroups generated by idempotents. Fund. Math. 84, 75–78 (1974)
15. Paseka, J., Rosicky, J.: Quantales. In: Coecke, B., Moore, D., Wilce, A. (eds.) Current Research in Operational Quantum Logic: Algebras. Categories and Languages. Fund. Theories Phys., vol. 111, pp. 245–262. Kluwer Academic Publishers, Dordrecht (2000)
16. Birkhoff, G.: Lattice Theory. American Mathematical Society, Providence, RI (1967)
17. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1990)
18. Kuroda, S.Y.: Classes of languages and linear bounded automata. Information and Control 7, 203–223 (1964)
19. Spencer-Brown, G.: Laws of Form. Julian Press and Bantam, New York (1972)
20. Wood, D.: The Theory of Computation. Harper and Row, New York (1987)

A Grammars in the Algebraic Approach

The generalization of grammars to arbitrary monoids is, for the most part, straightforward. However, there are few elements which require further elaboration.

Classically, a grammar over the alphabet X affixes a set Q of indeterminates, called either *variables* or (making reference to the notion of parse trees) *non-terminals*. It is assumed that $X \cap Q = \emptyset$. A finite set H of schemes is provided for effecting transitions over configurations in $(X \cup Q)^*$, so that

$$H \subseteq (X \cup Q)^* \times (X \cup Q)^* .$$

A *starting configuration* $S \in (X \cup Q)^*$ is identified and the language is defined as the set of all the words in X^* derivable from S by a finite number of applications of transitions $u \rightarrow v$ for $(u, v) \in H$ to subwords in the present configuration. One usually assumes the starting configuration $S \in Q$ to be one of the variables, though this restriction is not essential.

When generalizing to an arbitrary monoid M , one may assume that $X \subseteq M$ is a distinguished subset, though its explicit delineation does not prove to be essential. In place of $(X \cup Q)^*$, one must then take the *free extension* $M[Q]$

of the monoid M by the indeterminates in Q . In the case where $M = X^*$ and $X \cap Q = \emptyset$, the free extension (see below) reduces (up to isomorphism) to $M[Q] \cong (X \cup Q)^*$.

A.1 Free Extensions of Monoids

Thus, in its more general form, a grammar is a structure $G = (Q, S, H)$ over a monoid M composed of a set of variables, Q ; a distinguished configuration $S \in M[Q]$; and a set of transition rules $H \subseteq M[Q] \times M[Q]$. In the grammars we consider, H will always be finite. This definition includes, as a special case where $M = X^* \times Y^*$, *translations* from X to Y ; and $M = X^*$, for *languages* over alphabet X . More interesting examples might be conceived of where M represents a construction language for graphical or multimedia displays (e.g. a typesetting, hypertext or word processing language); for instance, the commutative monoid that underlies the 2-dimensional symbolic language used in the *Laws of Form* [19] for Boolean algebra.

The monoid $M[Q]$ is the free extension of M by the set Q . It may be thought of as the monoid M , itself, with the set Q of indeterminates added to it. A word $\alpha \in M[Q]$ may be written as $\alpha = m_0 q_1 m_1 \dots q_n m_n$, its degree being $\deg(\alpha) = n \geq 0$. The monoid product is defined by

$$(m_0 q_1 m_1 \dots q_n m_n) (n_0 r_1 n_1 \dots r_n n_p) \equiv m_0 q_1 m_1 \dots q_n (m_n n_0) r_1 n_1 \dots r_n n_p,$$

with $\deg(\alpha\beta) = \deg(\alpha) + \deg(\beta)$. Classically, one has $M = X^*$ and $M[Q] = (X^*)[Q] = (X \cup Q)^*$, provided that $X \cap Q = \emptyset$. The identity is just the monoid identity $1 \in M$. The monoid M is embedded within $M[Q]$ as the words of degree 0, while the set Q is mapped to the words of degree 1 of the form $(1q1)$.

The free extension $M[Q]$ has the following universal property. Corresponding to a monoid homomorphism $\phi : M \rightarrow N$ and map $\sigma : Q \rightarrow N$ is a unique monoid homomorphism $\langle \phi, \sigma \rangle : M[Q] \rightarrow N$ such that $\langle \phi, \sigma \rangle(m) = \phi(m)$ for words $m \in M \subseteq M[Q]$ of degree 0, and $\langle \phi, \sigma \rangle(1q1) = \sigma(q)$. The map is uniquely given from these criteria by

$$\langle \phi, \sigma \rangle(m_0 q_1 m_1 \dots q_k m_k) = \phi(m_0) \sigma(q_1) \phi(m_1) \dots \sigma(q_k) \phi(m_k).$$

A.2 Generalized Grammars

A rule $(\alpha, \beta) \in H$ is then to be thought of as a *one-step transition* $\alpha \rightarrow \beta$. More generally, a transition sequence is a sequence of words in $M[Q]$ of the form $\alpha_0 \rightarrow \dots \rightarrow \alpha_n$, where $n \geq 0$, such that adjacent members of the sequence are of the form $\gamma\alpha\delta = \gamma\beta\delta$ for some $\gamma, \delta \in M[Q]$, $(\alpha, \beta) \in H$. Corresponding to each $\alpha \in M[Q]$ is the subset $[\alpha] \equiv \{m \in M : \alpha \rightarrow m\}$ of elements of M derivable from the configuration α . The language $L(G) \subseteq M$ corresponding to the grammar is that $L(G) \equiv [S] = \{m \in M : S \rightarrow m\}$ associated with the starting configuration.

Of particular interest are those grammars where H is restricted to the form $H \subseteq Q \times M[Q]$. Such a grammar is deemed to be *context-free*.

The family $\mathcal{CM}$ of *context-free* subsets of a monoid M shall consist of subsets $L(G) \subseteq M$ generated by a context-free grammar $G = (Q, S, H)$, for finite H . Similarly, the family $\mathcal{TM}$ may be defined, where G is a general grammar (again, with H being finite.)

A.3 Normed Monoids and Context-Sensitive Grammars

A question arises as to how to define *context-sensitive* subsets for monoids other than free monoids $M = X^*$. The classical definition makes explicit reference to the *length* of the elements of $(X \cup Q)^*$, requiring that a restriction be placed on $H \subseteq Q^* \times (X \cup Q)^*$, such that $0 < \ln(\alpha) \leq \ln(\beta)$, for each $(\alpha, \beta) \in H$; where $\ln(\alpha)$ denotes the length of the word $\alpha \in (X \cup Q)^*$. One-step derivations should be restricted to a form where only variables should appear on the left, and the right-hand side should be of length no less than that of the left. Thus, we are able to define a class $\mathcal{SX}^*$ comprising the *context-sensitive* subsets of the free monoid X^* .⁸

A generalization to arbitrary monoids may be found if we require that the monoid operator $M \mapsto SM$ be well-behaved under monoid homomorphisms. In particular, if $X \subseteq M$ is a generating subset of the monoid M then under the canonical homomorphism $\sigma_X : X^* \rightarrow M$, we should expect that $SM = \{\sigma_X(A) : A \in \mathcal{SX}^*\}$. First, in order for the length restriction to be satisfied, we should require that $1 \notin X$. Second, in order for the definition to be well-behaved, we should also require independence of the selection of a generating subset. In particular, if $Y \subseteq M$ is any other generating subset of M such that $1 \notin Y$, with a canonical homomorphism $\sigma_Y : Y^* \rightarrow M$, then there should be a way to convert a context-sensitive grammar $G = (Q, S, H)$ over X^* to one $G' = (Q', S', H')$ over Y^* . Indeed, this can be done by adding new variables $\hat{x}$ for each $x \in X$, replacing each symbol from X in the original grammar with the corresponding variable, and then adding new rules $\hat{x} \mapsto w_x$, where $\sigma_Y(w_x) = \sigma_X(x)$. That is, we define $Q' = Q \cup \{\hat{x} : x \in X\}$, $S' = h(S)$ and

$$H' = \{(h(\alpha), h(\beta)) : (\alpha, \beta) \in H\} \cup \{(\hat{x}, w_x) : x \in X\},$$

where $h : (X \cup Q)^* \rightarrow Q'^*$ is the monoid homomorphism defined inductively by $h(x) = \hat{x}$, for $x \in X$ and $h(q) = q$, for $q \in Q$. It is not too difficult, then, to show that $\alpha \rightarrow \beta$ in G if, and only if, $h(\alpha) \rightarrow h(\beta)$ in G' and that $\sigma_X([a]_G) = \sigma_X([h(a)]_{G'})$, for $\alpha \in (X \cup Q)^*$. The length requirement is also satisfied since $\ln(h(\alpha)) = \ln(\alpha)$ and $1 = \ln(\hat{x}) \leq \ln(w_x)$. The latter property is where we specifically require that $1 \notin X$.

⁸ This variety of grammar is known, in classical theory, as the *monotonic* or *non-contracting* grammar. A *context-sensitive* grammar, classically, admits rules of the form $\alpha q \beta \rightarrow \alpha \gamma \beta$, with the restriction $q \in Q$. This allows production of the empty word, whereas monotonic grammars do not. Therefore, explicit stipulation must be made to allow for the inclusion of the monoid identity $1 \in M$ in the members of the family SM .

The central feature of the context-sensitivity concept is the notion of length. This is what we are actually generalizing to arbitrary monoids. Each generating subset $X \subseteq M$ defines a length function such that the elements of X have minimal length. This leads naturally to the following definitions:

Definition 5 (Normed Monoids). *M is a monoid with a length function $m \in M \mapsto \|m\| \in \mathbb{R}$ such that*

Non-negativity $\|m\| \geq 0$, for $m \in M$,

Non-degeneracy $\|m\| = 0 \leftrightarrow m = 1$ for $m \in M$,

Triangle inequality $\|mm'\| \leq \|m\| + \|m'\|$ for $m, m' \in M$.

An element $m \in M - \{1\}$ is atomic with respect to the norm if

$$m = m_1 m_2 \rightarrow m_1 = 1 \vee m_2 = 1 \vee \|m\| < \|m_1\| + \|m_2\|.$$

If $\inf_{x \in X} \|x\| > 0$, where X denotes the set of atomic elements, then the norm will be called atomic.

It follows, by a routine induction, that the atomic elements X corresponding to an atomic norm comprise a generating subset of the monoid M . Conversely, given a generating subset $X \subseteq M - \{1\}$, we may define the length by $\|1\|_X = 0$, $\|m\|_X = n + 1$ for $(m \in \sigma_X(X^{n+1}) - \sigma_X(X^n))$.

A norm over the monoid M may be extended to a norm over $M[Q]$ by defining $\|q\| = 1$ for $q \in Q$. It will then follow that $X \cup Q$ will comprise the corresponding set of atomic elements. Moreover, the property of atomicity will be preserved by the norm.

The context-sensitive grammar over M is then a "value-reducing" grammar with respect to a given norm; that is, a grammar whose one-step derivations are restricted to the form $(\alpha, \beta) \in H$ where $0 < \|\alpha\| \leq \|\beta\|$, with the prescription that α consist only of variables.

The Algebraic Approach II: Dioids, Quantales and Monads

Mark Hopkins

The Federation Archive
federation2005@netzero.net
<http://federation.g3z.com>

Abstract. The algebraic approach to formal language and automata theory is a continuation of the earliest traditions in these fields which had sought to represent languages, translations and other computations as expressions (e.g. regular expressions) in suitably-defined algebras; and grammars, automata and transitions as relational and equational systems over these algebras that have such expressions as their solutions.

As part of a larger programme to algebraize the classical results of formal language and automata theory, we have recast and generalized the Chomsky hierarchy as a complete lattice of dioid algebras. Here, we will formulate a general construction by ideals that yields a family of adjunctions between the members of this hierarchy.

In addition, we will briefly discuss the extension of the dioid hierarchy to semirings and power series algebras.

Keywords: Monad, Ideal, Adjunction, Category, Dioid, Semiring, Quantale, Kleene.

1 Preliminaries

1.1 The Algebraic Point of View

In the standard formulation of formal languages and automata, which we will refer to henceforth as the classical theory, a language is usually regarded as a subset of a free monoid $M = X^*$. In contrast, in the *Algebraic Approach*, a formal language is viewed as an algebraic entity residing in a partially ordered monoid. Through the conventional identification $x \leftrightarrow \{x\}$, the point of view grounded in set theory is algebraized, with each set actually being viewed as a *sum* of its elements, e.g.,

$$\{a^m b^m : m \geq 0\} = \bigcup_{m \geq 0} \{a\}^m \{b\}^m \leftrightarrow \sum_{m \geq 0} a^m b^m.$$

In the classical theory, the process of algebraization ended abruptly at the type 3 level in the Chomsky hierarchy: the regular languages and their corresponding algebra of regular expressions. Attempts were made to extend this process to the type 2 level (i.e., context-free expressions) [23,4], but did not find particularly

fruitful applications; e.g., no algebraic reformulation of parsing theory. A significant step, however, in this direction had already been taken early on [12], the result being the Chomsky-Schützenberger Theorem for context-free languages. However, no theory of context-free expressions arose from this result. In recent times, we've begun to see renewed progress in this direction [5].

Much of what stood in the way may have been the difficulty in clarifying the algebraic foundation underlying the theory of regular expressions. In what algebra(s) do these objects live? A diversity of answers emerged, as had been noted in [8], in which adjunctions were constructed to embody the hierarchical relation $\mathcal{R} \leq \omega \leq \mathcal{P}$.

Though the large number of inequivalent formulations may seem to be a setback, in fact, as we have seen in [1], a complete lattice of monadic dioids can be defined which presents no less than an embodiment and significant generalization of the Chomsky Hierarchy, itself. For, in addition to the operators $\mathcal{F}M$, $\mathcal{R}M$, ωM and $\mathcal{P}M$ defining, respectively, the finite, rational, countable and general subsets of a monoid M , we also have operators $\mathcal{C}M$, $\mathcal{S}M$ and $\mathcal{T}M$ defining, respectively, the context-free, context-sensitive and Turing-computable subsets of M . Correspondingly, one may then seek to define adjunctions between the members of the larger hierarchy

$$\mathcal{F} \leq \mathcal{R} \leq \mathcal{C} \leq \mathcal{S} \leq \mathcal{T} \leq \omega \leq \mathcal{P}$$

and, indeed, between all the members of the monadic dioid lattice, itself.

A precursor to the results formulated here may be found in [8], where adjunctions are defined connecting the operators $\mathcal{R} \leq \omega \leq \mathcal{P}$ and their corresponding categories of dioids, which we shall term $\mathbf{DR}$, $\mathbf{D}\omega$ and $\mathbf{DP}$. The functors

$$\mathbf{DR} \rightarrow \mathbf{D}\omega \rightarrow \mathbf{DP}$$

are constructed by defining appropriate families of ideals for the respective algebras, while the opposite members of the respective adjoint pairs

$$\mathbf{DR} \leftarrow \mathbf{D}\omega \leftarrow \mathbf{DP}$$

give us the structure-reducing forgetful functors. Conway [9] had earlier provided a construction for the adjunction formed of the pair $\mathbf{Q}_{\mathcal{R}}^{\mathcal{P}} : \mathbf{DR} \rightarrow \mathbf{DP}$, and $\mathbf{Q}_{\mathcal{P}}^{\mathcal{R}} : \mathbf{DP} \rightarrow \mathbf{DR}$.

These constructions may also be viewed as results in Kleene algebra, whereby a given $*$ -continuous Kleene algebra is extended to a form that has closure and distributivity under a larger family of subsets. Expanding on this point of view, the adjunctions relating the pairs $\mathcal{R} \leq \mathcal{C}$, $\mathcal{R} \leq \mathcal{S}$ and $\mathcal{R} \leq \mathcal{T}$ may be viewed as operations that give us a fixed-point closure of a given $*$ -continuous Kleene algebra for $\mathcal{C}$, or a relational closure for $\mathcal{S}$ and $\mathcal{T}$. Concrete realizations of these constructions, in particular for $\mathcal{C}$, would then provide us with an algebraization of the classical result known as the Chomsky-Schützenberger theorem (thus, also resolving a question raised in the closing section of [5]).

More generally, denoting by $\mathbf{DA}$ the category of $\mathcal{A}$ -dioids and $\mathcal{A}$ -morphisms¹, a desired outcome would be to reflect the hierarchy of monadic dioids by a hierarchy of adjunctions $\mathbf{Q}_A^B : \mathbf{DA} \rightarrow \mathbf{DB}$ where $A \leq B$, such that $\mathbf{Q}_B^C \circ \mathbf{Q}_A^B = \mathbf{Q}_A^C$, whenever $A \leq B \leq C$.

1.2 Monadic Operators

Different families of languages over an alphabet X are defined through their corresponding families of subsets of a monoid X^* . When expressed the algebraic setting formulated in [1], each family is identified as a monadic operator, $\mathcal{A} : M \mapsto \mathcal{A}M$ that freely extends a monoid M to an $\mathcal{A}$ -dioid. Reviewing the basic results of [1], such an operator, $\mathcal{A}M$, $(\mathbf{A}_1)$ is defined as a family of subsets of the monoid M ; $(\mathbf{A}_2)$ contains all the finite subsets of M ; $(\mathbf{A}_3)$ is closed under products (thus making $\mathcal{A}M$ a monoid); $(\mathbf{A}_4)$ is closed under unions from $\mathcal{A}M$; and $(\mathbf{A}_5)$ respects homomorphisms in the sense that if $f : M \rightarrow N$ is a monoid homomorphism, then² $\tilde{f}(U) \in \mathcal{A}N$ for all $U \in \mathcal{A}M$. Though property $\mathbf{A}_3$ is not a part of the classical theory, we are able to prove the equivalence of the combination of $\mathbf{A}_3$ and $\mathbf{A}_4$ with a property that is classical: that $(\mathbf{A}_5)$ $\mathcal{A}$ respects $\mathcal{A}$ -substitutions³ – if $\sigma : M \rightarrow \mathcal{P}N$ is an $\mathcal{A}$ -substitution, then $\hat{\sigma}(U) \in \mathcal{A}N$ for all $U \in \mathcal{A}M$. Finally, we are able to show, given the surjectivity of the monoid homomorphism $f : M \rightarrow N$, the surjectivity of its lift $\tilde{f} : \mathcal{A}M \rightarrow \mathcal{A}N$ (property $\mathbf{A}_6$, [1]).

Using the notation $x > A$ to denote when x is an upper bound of a set A , one may then define M to be $(\mathbf{D}_0)$ $\mathcal{A}$ -additive if every $U \in \mathcal{A}M$ has a least upper bound $\sum U \in M$; $(\mathbf{D}_1)$ $\mathcal{A}$ -separable if for all $x > aUb$ there exists $x > U$ such that $x \geq aub$, where $a, b \in M$ and $U \in \mathcal{A}M$; and $(\mathbf{D}_2)$ strongly $\mathcal{A}$ -separable if for all $x > UV$ there exist $u > U, v > V$ such that $x \geq uv$, where $U, V \in \mathcal{A}M$. Finally, a monoid homomorphism $f : M \rightarrow N$ is said to be $(\mathbf{D}_3)$ $\mathcal{A}$ -continuous if for all $y > \tilde{f}(U)$ there exists $x > U$ such that $y \geq f(x)$, where $U \in \mathcal{A}M$.

When a monoid is $\mathcal{A}$ -additive both forms of separability equivalently reduce to more familiar form as the following distributivity identities: $(\mathbf{D}_{1'})$ $a, b \in M, U \in \mathcal{A}M \rightarrow \sum (aUd)$; and $(\mathbf{D}_{2'})$ $U, V \in \mathcal{A}M \rightarrow \sum (UV) = \sum U \cdot \sum V$. Also, in such monoids, for order-preserving monoid homomorphisms, $f : M \rightarrow M'$, $\mathcal{A}$ -additivity reduces equivalently to the condition that $(\mathbf{D}_{3'})$: $U \in \mathcal{A}M \rightarrow f(\sum U) = \sum \tilde{f}(U)$. Finally, an $\mathcal{A}$ -dioid is a partially ordered monoid M satisfying $\mathbf{D}_0$ and $\mathbf{D}_1$, and an $\mathcal{A}$ -morphism is an order-preserving monoid homomorphism that satisfies $\mathbf{D}_3$.

The following results may then be proven:

Theorem 1 (*The Universal Property, [1]*). *The free $\mathcal{A}$ dioid extension of a monoid M is $\mathcal{A}M$. Equivalently, this may be stated as follows: that $\eta_M : M \rightarrow$*

¹ Defined in [1], these will be reviewed here in the following section.

² In here, and in the following, we will denote the image of a function f on a set U by $\tilde{f}(U) \equiv \{f(u) : u \in U\}$.

³ Recalling [1], an $\mathcal{A}$ -substitution is a monoid homomorphism $\sigma : M \rightarrow \mathcal{A}N$ and is uniquely determined by its extension $\hat{\sigma} : \mathcal{P}M \rightarrow \mathcal{P}N$, given by $\hat{\sigma} : U \subseteq M \mapsto \bigcup_{u \in U} \sigma(u)$ to a (unit-preserving) quantale homomorphism [1].

AM , $m \mapsto \{m\}$ is a monoid homomorphism and that a monoid homomorphism $f : M \rightarrow D$ to an $\mathcal{A}$ -doid D extends uniquely to an $\mathcal{A}$ -morphism $f^* : AM \rightarrow D$; i.e., such that $f = f^* \circ \eta_M$.

Theorem 2 (Hierarchical Completeness, [1]). *Monadic operators form a complete lattice with top $AM = \mathcal{P}M$ and bottom $AM = \mathcal{F}M$, with lattice meet defined for a family Z be a family of monadic operators by $(\wedge Z)M = \bigcap_{A \in Z} AM$.*

We will use $\geq$ and $\leq$ to denote the lattice ordering relation, $\mathcal{A} \leq \mathcal{B} \leftrightarrow \mathcal{A} \cap \mathcal{B} = \mathcal{A}$ and $\mathcal{B} \geq \mathcal{A} \leftrightarrow \mathcal{A} \leq \mathcal{B}$.

2 Ideals and Quantaes

In this section, we will define the quantale completion for each variety of monadic doid. The construction is accomplished through a suitably defined family of ideals and is similar to that used to define the completion of a lattice.

2.1 Ideals, Basic Properties

Corresponding to each operator $\mathcal{A}$ is a variety of ideals that will be termed $\mathcal{A}$ -ideals. The definition makes use of the following closure, which is generic to partial orderings.

Definition 1. *For a partially ordered set D , let $U' = \{x \in D : \forall y > U : y \geq x\}$.*

If a set U has a least upper bound $\sum U$, then the relation $y > U$ is equivalent to $y \geq \sum U$. Therefore, defining the *interval* $\langle a \rangle \equiv \{x \in D : x \leq a\}$ we have the following properties.

Theorem 3. *For a partially ordered set D :*

- $\{a\}' = \langle a \rangle$;
- if 0 is the minimal element of D , then $\emptyset' = \{0\}$; and
- if $U \subseteq D$ has a least upper bound $\sum U$ then $U' = \langle \sum U \rangle$.

We may define the family $\mathcal{A}[D]$ of $\mathcal{A}$ -ideals in the general setting of partially ordered monoids, D . The sole requirement we impose on such ideals $I \subseteq D$ is that **(I₁)**: for all $U \in \mathcal{A}D$ and $a, b \in D$, if $aUb \subseteq I$, then $aU'b \subseteq I$.

Since $\{a\}' = \langle a \rangle$, property **I₁** implies that an $\mathcal{A}$ -ideal I must also be closed downward with respect to the partial ordering $\leq$, **(I₂)**: $x \leq d \in I \rightarrow x \in I$.

Though the definition is generic to partially ordered monoids, its primary application will be to $\mathcal{A}$ -doids, D . In such a case, an $\mathcal{A}$ -ideal of D may be equivalently defined by property **(I₃)**: $U \in \mathcal{A}D$, $U \subseteq I \rightarrow \sum U \in I$. We prove this in the following.

Corollary 1. *For $\mathcal{A}$ -doids, D , **I₁** is equivalent to **I₂** and **I₃**.*

Proof. Taking $a = b = 1$ in $\mathbf{I}_1$, leads to the result $\mathbf{I}_3$. For the converse, we note that the $\mathcal{A}$ -separability property $\mathbf{D}_1$ of D implies for $U \in \mathcal{A}D$ and $a, b \in D$ that

$$a \left\langle \sum U \right\rangle b = aU'b \subseteq (aUb)' = \left\langle \sum (aUb) \right\rangle'.$$

Combined with $\mathbf{I}_2$ and $\mathbf{I}_3$, this leads to $\mathbf{I}_1$.

For $\mathcal{A} = \mathcal{F}, \mathcal{R}$, equivalent definitions of $\mathcal{A}$ -ideals may be formulated in the general setting of dioids. In particular, since $\emptyset' = \{0\}$, property $\mathbf{I}_1$ requires that $0 \in I$.

Corollary 2. *Let D be a dioid. Then for an $\mathcal{A}$ -ideal $I \subseteq D$,*

IF₀ $I \neq \emptyset$;

IF₁ $0 \in I$;

IF₂ $d, e \in I \rightarrow d + e \in I$.

Moreover, an $\mathcal{F}$ -ideal $I \subseteq D$ is equivalently defined by $\mathbf{I}_2$, **IF₁** (or, equivalently, **IF₀**) and **IF₂**.

Proof. All three properties **IF₀**, **IF₁** and **IF₂** follow from $\mathbf{I}_3$, for the case $\mathcal{A} = \mathcal{F}$. Taking $a = b = 1$ with $U = \emptyset$ yields **IF₁** from which **IF₀** follows, while taking $a = b = 1$ with $U = \{d, e\}$ yields **IF₂**. Similarly, for $\mathcal{A}$ -dioids, the result follows in virtue of the inclusion $\mathcal{F}D \subseteq \mathcal{A}D$.

Conversely, suppose $\mathbf{I}_2$, **IF₁** and **IF₂** hold, and that $U = \{u_1, \dots, u_n\} \subseteq D$ with $n \geq 0$. Then we have $n = 0 \rightarrow \sum U = \sum \emptyset = 0 \in I$ by **IF₁** and $\mathbf{I}_2$, and $n > 0 \rightarrow \sum U = u_1 + \dots + u_n \in I$ by **IF₂**.

For the operator $\mathcal{R}$, we have the following characterization:

Corollary 3. *An $\mathcal{R}$ -ideal $I \subseteq D$ of an $\mathcal{R}$ -dioid D is an $\mathcal{F}$ -ideal of D for which:*

IR₁ *if $ab^n c \in I$ for all $n \geq 0$, then $ab^*c \in I$.*

Proof. If $I \subseteq D$ is an $\mathcal{R}$ -ideal, from $a\{b\}^*c = \{ab^n c : n \geq 0\} \subseteq I$, we conclude that $ab^*c = \sum a\{b\}^*c \in I$, by $\mathbf{I}_3$. To prove the converse, for an $\mathcal{F}$ -ideal $I \subseteq D$ satisfying **IR₁**, we need to inductively establish, for $U \in \mathcal{R}D$, that $aUd \subseteq I \rightarrow a(\sum U)d \in I$. The argument is quite analogous to that used to establish the equivalence of $\mathcal{R}$ -dioids and $*$ -continuous Kleene algebras. We already have the property for finite subsets, by assumption. Showing that the property is preserved by sums, products, stars is easy, noting the following

$$\begin{aligned} a \left(\sum U \cup V \right) d &= a \left(\sum U \right) d + a \left(\sum V \right) d, \\ a \left(\sum UV \right) d &= a \left(\sum U \right) \left(\sum V \right) d = \sum_{v \in V} a \left(\sum U \right) vd, \\ a \left(\sum U^* \right) d &= \sum_{n \geq 0} a \left(\sum U \right)^n d \end{aligned}$$

and using **IR₁** in conjunction with the last equality.

In general $\mathcal{A}$ -ideals will form a hierarchy closed under intersection. This is a consequence of the following:

Theorem 4. *For a partially ordered monoid D , $Y \subseteq \mathcal{A}[D] \rightarrow \cap Y \in \mathcal{A}[D]$.*

Proof. Let $Y \subseteq \mathcal{A}[D]$. Then suppose $U \in \mathcal{A}D$ with $aUb \subseteq \cap Y$. Then for any $\mathcal{A}$ -ideal $I \in Y$, we have $aUb \subseteq \cap Y \subseteq I \rightarrow aU'b \subseteq I$, by **I**₁. Hence, $aU'b \subseteq \cap Y$, thus making Y an $\mathcal{A}$ -ideal too. For the special case where $Y = \emptyset$, we set $\cap \emptyset = D$ and note that D is an ideal of itself.

As a result it follows that $\mathcal{A}[D]$ forms a complete lattice under the subset ordering $\subseteq$ with D as the maximal element. One may therefore define the ideal-closure of arbitrary sets:

Definition 2. *Let D be a partially ordered monoid and $U \subseteq D$. Then $\langle U \rangle_{\mathcal{A}} = \cap \{I \in \mathcal{A}[D] : U \subseteq I\}$.*

Basic properties, generic to partially ordered monoids, include the following:

Theorem 5. *In any partially ordered monoid D , if $U, V \subseteq D$ then*

$$\begin{aligned} U &\subseteq \langle U \rangle_{\mathcal{A}}, \\ U \subseteq V &\rightarrow \langle U \rangle_{\mathcal{A}} \subseteq \langle V \rangle_{\mathcal{A}}, \\ U \in \mathcal{A}[D] &\leftrightarrow U = \langle U \rangle_{\mathcal{A}}. \end{aligned}$$

For brevity, in the following we will usually omit the index and just write $\langle U \rangle$ for $\langle U \rangle_{\mathcal{A}}$, where the context permits. In the special case of $\mathcal{A}$ -diods, the following results also hold:

Corollary 4. *Let D be an $\mathcal{A}$ -diod. Then $\langle \emptyset \rangle = \langle 0 \rangle$ is the minimal $\mathcal{A}$ -ideal in D ; and each interval $\langle a \rangle = \langle \{a\} \rangle$, for $a \in D$, is a principal $\mathcal{A}$ -ideal in D .*

More generally, if D is already an $\mathcal{A}$ -diod, then $U' = \langle \sum U \rangle$ for any $U \in \mathcal{A}D$, so that these subsets generate principal ideals.

Lemma 1. *Let D be an $\mathcal{A}$ -diod. Then for any $U \in \mathcal{A}D$, then $\langle U \rangle_{\mathcal{D}} = \langle \sum U \rangle$.*

This then shows that the ideals generated by the subsets from $\mathcal{A}D$ will be in a one-to-one correspondence with D itself, when D has the structure of an $\mathcal{A}$ -diod. Taking the ideals generated from a larger family $\mathcal{B}D$ provides the natural candidate for the extension of D to a $\mathcal{B}$ -diod. If we could define the product and sum operations on ideals, then this would provide a basis for extending the $\mathcal{A}$ -diod D to a $\mathcal{B}$ -diod for an operator $\mathcal{B} > \mathcal{A}$. We would simply take those ideals generated from $\mathcal{B}D$.

In the most general case, where $\mathcal{B} = \mathcal{P}$, the family of ideals generated is just $\mathcal{A}[D]$, itself. The entire collection of ideals should then yield a full-fledged quantale structure. In fact, this is what we will examine next.

2.2 Defining a Quantale Structure on Ideals

The family $\mathcal{A}[D]$, when provided with a suitable algebraic structure, will define the extension of D to a dioid with the structure characteristic of a $\mathcal{P}$ -dioid or quantale with identity 1: a complete upper semilattice in which distributivity applies to all subsets. As a result, we will be able to define the map $\mathbf{Q}_{\mathcal{A}} : D \rightarrow \mathcal{A}[D]$ that yields a functor $\mathbf{Q}_{\mathcal{A}} : \mathbf{DA} \rightarrow \mathbf{DP}$ from the category $\mathbf{DA}$ of $\mathcal{A}$ -dioids and $\mathcal{A}$ -morphisms to the category $\mathbf{DP}$ of quantales (with units) and quantale (unit-preserving) morphisms.

Products. The product of two ideals should preserve the correspondence $\langle U \rangle = \langle \sum U \rangle$ that holds in $\mathcal{A}$ -dioids D with respect to $\mathcal{A}$ -ideals generated by subsets from $\mathcal{A}D$. But this would require that

$$\langle \sum U \sum V \rangle \leftrightarrow \langle UV \rangle \leftrightarrow \langle \sum UV \rangle.$$

Therefore, the product should satisfy the property $\langle U_1 V_1 \rangle_{\mathcal{A}} = \langle U_2 V_2 \rangle_{\mathcal{A}}$ whenever $\langle U_1 \rangle_{\mathcal{A}} = \langle U_2 \rangle_{\mathcal{A}}$ and $\langle V_1 \rangle_{\mathcal{A}} = \langle V_2 \rangle_{\mathcal{A}}$. We will prove this is so by showing, in particular, the following result. For brevity, we will again omit the subscript.

Lemma 2 (The Product Lemma). *Suppose D is a dioid and that $U, V \subseteq D$. Then $\langle UV \rangle = \langle \langle U \rangle \langle V \rangle \rangle$.*

Proof. One direction is already immediate: from $U \subseteq \langle U \rangle$ and $V \subseteq \langle V \rangle$, we get $UV \subseteq \langle U \rangle \langle V \rangle$. Consequently, $\langle UV \rangle \subseteq \langle \langle U \rangle \langle V \rangle \rangle$. In the other direction, if we can show that $\langle U \rangle \langle V \rangle \subseteq \langle UV \rangle$ then it will follow that

$$\langle \langle U \rangle \langle V \rangle \rangle \subseteq \langle \langle UV \rangle \rangle = \langle UV \rangle.$$

To this end, let $Y = \{y \in D : yV \subseteq \langle UV \rangle\}$ and $Z = \{z \in D : \langle U \rangle z \subseteq \langle UV \rangle\}$. Then clearly $YV \subseteq \langle UV \rangle$ and $U \subseteq Y$. So, if we can show that Y is an ideal, it will then follow that $\langle U \rangle \subseteq \langle Y \rangle = Y$, from which we could conclude $\langle U \rangle V \subseteq \langle UV \rangle$. From this, in turn, it will follow that $V \subseteq Z$, while $\langle U \rangle Z \subseteq \langle UV \rangle$. So, if we can also show that Z is an ideal, then we will be able to conclude that $\langle V \rangle \subseteq \langle Z \rangle = Z$ and, from this, that $\langle U \rangle \langle V \rangle \subseteq \langle U \rangle Z \subseteq \langle UV \rangle$.

Suppose, then, that $aWb \subseteq Y$, where $a, b \in D$ and $W \in \mathcal{A}D$. Then, for each $v \in V$, by definition of Y , we have $aWbv \subseteq \langle UV \rangle$. Applying property **I**₁ to the ideal $\langle UV \rangle$, we conclude that $aW'bv \subseteq \langle UV \rangle$. Therefore, it follows that $aW'bV \subseteq \langle UV \rangle$ and, from this, that $aW'b \subseteq Y$. Thus, Y is an ideal.

The argument showing that Z is an ideal is similar. Suppose $aWb \subseteq Z$, again, with $a, b \in D$ and $W \in \mathcal{A}D$. Then, for each $u \in \langle U \rangle$, by definition of Z , we have $uaWb \subseteq \langle UV \rangle$. Again applying property **I**₁ to the ideal $\langle UV \rangle$, we conclude that $uaW'b \subseteq \langle UV \rangle$, from this it follows that $\langle U \rangle aW'b \subseteq \langle UV \rangle$ and $aW'b \subseteq Z$.

This clears the way for us to define products over subsets of D .

Definition 3. *Let D be a dioid, and $U, V \subseteq D$. Then define $U \cdot V \equiv \langle UV \rangle$.*

Lemma 3. *Let D be a dioid. Then $\mathcal{A}[D]$ is a partially ordered monoid with product $U, V \mapsto U \cdot V$, identity $\langle \{1\} \rangle$ and ordering $\subseteq$.*

Proof. Let $U, V, W \subseteq D$ Then

$$\langle \{1\} \rangle \cdot \langle V \rangle = \langle \{1\} V \rangle = \langle V \rangle = \langle V \{1\} \rangle = \langle V \rangle \cdot \langle \{1\} \rangle ,$$

$$\langle U \rangle \cdot (\langle V \rangle \cdot \langle W \rangle) = \langle U \rangle \cdot \langle VW \rangle = \langle UVW \rangle = \langle UV \rangle \cdot \langle W \rangle = (\langle U \rangle \cdot \langle V \rangle) \cdot \langle W \rangle .$$

We can treat this algebra as an inclusion of the monoid structure of D , itself, through the correspondence $x \leftrightarrow \langle x \rangle$. But in general, it will not be an embedding, unless D also possesses the structure of an $\mathcal{A}$ -dioid. This result is captured by the following property:

Theorem 6. *If D is a $\mathcal{A}$ -dioid, then for $a, b \in D$, $\langle a \rangle \cdot \langle b \rangle = \langle ab \rangle$. Thus, $\langle \rangle_{\mathcal{A}} : D \rightarrow \mathcal{D}[D]$, is a monoid embedding with the unit $\langle 1 \rangle$.*

Proof. This follows from the relation between principal ideals and intervals, which generally holds in dioids:

$$\langle a \rangle \cdot \langle b \rangle = \langle \{a\} \rangle_{\mathcal{A}} \cdot \langle \{b\} \rangle_{\mathcal{A}} = \langle \{a\} \{b\} \rangle_{\mathcal{A}} = \langle \{ab\} \rangle_{\mathcal{A}} = \langle ab \rangle .$$

The one-to-one ness of $a \mapsto \langle a \rangle$ is a consequence of the anti-symmetry property of partial orders.

Sums. In a similar way, we would like to preserve the correspondence $U \leftrightarrow \langle \sum U \rangle$ with respect to the sum operator. So, if $U \in \mathcal{AD}$, then we should be able to express $\langle U \rangle_{\mathcal{A}}$ as a sum over its component principal ideals, $\langle U \rangle_{\mathcal{A}} = \sum_{u \in U} \langle u \rangle = \bigcup_{u \in U} \langle u \rangle$. In order for this to work, we need to know that if $\langle U_{\alpha} \rangle_{\mathcal{A}} = \langle V_{\alpha} \rangle_{\mathcal{A}}$ for all $\alpha \in A$, then $\langle \bigcup_{\alpha \in A} U_{\alpha} \rangle_{\mathcal{A}} = \langle \bigcup_{\alpha \in A} V_{\alpha} \rangle_{\mathcal{A}}$. In particular, we will prove the following result (omitting the subscript again, for brevity):

Lemma 4 (*The Sum Lemma*). *Let D be a dioid and $\mathbf{Y} \subseteq \mathcal{PD}$. Then $\langle \bigcup \mathbf{Y} \rangle = \langle \bigcup_{V \in \mathbf{Y}} \langle V \rangle \rangle$.*

Proof. Unlike the *Product Lemma* (lemma 2), this result may be established directly without an inductive proof. Suppose $\mathbf{Y} \subseteq \mathcal{PD}$. For $V \in \mathbf{Y}$, we then have the following line of argumentation

$$V \in \mathbf{Y} \rightarrow V \subseteq \bigcup \mathbf{Y} \rightarrow \langle V \rangle \subseteq \langle \bigcup \mathbf{Y} \rangle .$$

here, we can continue and argue as follows

$$\bigcup_{V \in \mathbf{Y}} \langle V \rangle \subseteq \langle \bigcup \mathbf{Y} \rangle \rightarrow \left\langle \bigcup_{V \in \mathbf{Y}} \langle V \rangle \right\rangle \subseteq \langle \langle \bigcup \mathbf{Y} \rangle \rangle = \langle \bigcup \mathbf{Y} \rangle .$$

Going in the opposite direction, we have the inclusions $V \subseteq \langle V \rangle$, for each $V \in \mathbf{Y}$. Therefore,

$$\bigcup \mathbf{Y} \subseteq \bigcup_{V \in \mathbf{Y}} \langle V \rangle \rightarrow \langle \bigcup \mathbf{Y} \rangle \subseteq \left\langle \bigcup_{V \in \mathbf{Y}} \langle V \rangle \right\rangle .$$

This clears the way for us to define a summation operator over $\mathcal{PD}$.

Definition 4. Let D be a dioid and $Y \subseteq \mathcal{P}D$. Then, define $\sum Y \equiv \langle \bigcup Y \rangle$.

Theorem 7. Let D be a dioid. Then $Y \mapsto \sum Y$ is the least upper bound operator over $\mathcal{A}[D]$.

Proof. Suppose $Y \subseteq \mathcal{P}D$ and $I \in \mathcal{A}[D]$ and upper bound. That is, assume that $V \subseteq I$ for all $V \in Y$. Then it follows that

$$\bigcup Y \subseteq I \rightarrow \sum Y = \langle \bigcup Y \rangle \subseteq \langle I \rangle = I.$$

But clearly $\sum Y$ is, itself, an upper bound of Y . Indeed, for all $V \in Y$, we have

$$V \subseteq \bigcup Y \subseteq \langle \bigcup Y \rangle = \sum Y.$$

Therefore, $\sum Y$ is the least upper bound of Y .

We can also prove that the Σ operator is distributive.

Lemma 5. Let D be a dioid, $U, V \subseteq D$ and $Y \subseteq \mathcal{P}D$. Then $U \cdot \sum Y \cdot V = \sum_{W \in Y} U \cdot W \cdot V$.

Proof. This is a direct consequence of definition 4 and theorem 7 with

$$U \cdot \sum Y \cdot V = U \cdot \langle \bigcup Y \rangle \cdot V = \langle U (\bigcup Y) V \rangle = \left\langle \bigcup_{W \in Y} UWV \right\rangle$$

while

$$\sum_{w \in Y} U \cdot W \cdot V = \sum_{W \in Y} \langle UWV \rangle = \left\langle \bigcup_{W \in Y} UWV \right\rangle.$$

Quantale Structure. Finally, this leads to the result

Theorem 8. For any dioid D and monadic operator $\mathcal{A}$, $\mathcal{A}[D]$ is a quantale with a unit $\langle \{1\} \rangle$. Moreover, if D is an $\mathcal{A}$ -dioid, then the map $\mathbf{Q}_{\mathcal{A}} : D \rightarrow \mathcal{A}[D]$ is an $\mathcal{A}$ -morphism.

Proof. In general, the restriction of the map $\langle \rangle_{\mathcal{A}} : \mathcal{A}D \rightarrow \mathcal{A}[D]$ is an order-preserving monoid homomorphism since $\langle \{1\} \rangle_{\mathcal{A}} = \langle 1 \rangle$ and $\langle U \rangle_{\mathcal{A}} \cdot \langle V \rangle_{\mathcal{A}} = \langle UV \rangle_{\mathcal{A}}$. When the dioid D also happens to have the structure of an $\mathcal{A}$ -dioid, then the correspondence reduces to an embedding $\mathbf{Q}_{\mathcal{A}} : D \rightarrow \mathcal{A}[D]$ into the principal ideals of D , for in that case, we have $\langle U \rangle_{\mathcal{A}} = \langle \sum U \rangle$, for all $U \in \mathcal{A}D$. The result is then an extension of the $\mathcal{A}$ -dioid D to a quantale $\mathcal{A}[D]$.

Morphisms. Finally, we should have consistency with respect to $\mathcal{A}$ -morphisms $f : D \rightarrow D'$. In particular, we'd like to have the property that $\langle \tilde{f}(U) \rangle_{\mathcal{A}} = \langle \tilde{f}(V) \rangle_{\mathcal{A}}$ whenever $\langle U \rangle_{\mathcal{A}} = \langle V \rangle_{\mathcal{A}}$. This result, too, will be true. We will prove it in the following form (once again, omitting the subscript for brevity).

Lemma 6 (*The Morphism Lemma*). *Let D, D' be dioids and $f : D \rightarrow D'$ an $\mathcal{A}$ -morphism. Then for all $U \subseteq D$, $\langle \tilde{f}(U) \rangle = \langle \tilde{f}(\langle U \rangle) \rangle$.*

Proof. The forward inclusion is easy since

$$U \subseteq \langle U \rangle \rightarrow \tilde{f}(U) \subseteq \tilde{f}(\langle U \rangle) \rightarrow \langle \tilde{f}(U) \rangle \subseteq \langle \tilde{f}(\langle U \rangle) \rangle.$$

To prove the converse inclusion, define $X = \{x \in D : f(x) \in \langle \tilde{f}(U) \rangle\}$. Then X is an $\mathcal{A}$ -ideal. For if $V \in \mathcal{AD}$ and $a, b \in D$ with $aVb \subseteq X$, then $\tilde{f}(aVb) \subseteq \langle \tilde{f}(U) \rangle$. Since f is a monoid homomorphism, then $\tilde{f}(aVb) = f(a)\tilde{f}(V)f(b)$. Moreover, by property **A**₄, since $V \in \mathcal{AD}$, then $\tilde{f}(V) \in \mathcal{AD}'$. Therefore, applying **I**₁ to the ideal $\langle \tilde{f}(U) \rangle$, we get $f(a)\tilde{f}(V)'f(b) \subseteq \langle \tilde{f}(U) \rangle$. If we can then show that $f(V') \subseteq f(V)'$, then it will follow that

$$\tilde{f}(aV'b) = f(a)\tilde{f}(V')f(b) \subseteq f(a)\tilde{f}(V)'f(b) \subseteq \langle \tilde{f}(U) \rangle,$$

so that $aV'b \subseteq X$, thus proving that X is an ideal. With that given, then noting $U \subseteq X$, we would have $\langle U \rangle \subseteq \langle X \rangle = X$, and finally $\tilde{f}(\langle U \rangle) \subseteq \tilde{f}(X) \subseteq \langle \tilde{f}(U) \rangle$.

It is at this point that the $\mathcal{A}$ -additivity of f comes into play. Let $x \in V'$. Pick any upper bound $y > \tilde{f}(V)$. Then by the $\mathcal{A}$ -additivity of f , we have $y \geq f(v)$ for some upper bound $v > V$. By definition of V' , it then follows that $x \leq v$. In turn, by the order-preserving property of f (which is a part of the definition of an $\mathcal{A}$ -morphism), it follows that $f(x) \leq f(v) \leq y$. Thus, $f(x) \in \tilde{f}(V)'$.

This result clears the way to unambiguously defining the lifting of f to a mapping $f_{\mathcal{A}} : \mathcal{A}[D] \rightarrow \mathcal{A}[D']$ over the respective quantales.

Definition 5. *Let D, D' be dioids, and $f : D \rightarrow D'$ an $\mathcal{A}$ -morphism. Then define $f_{\mathcal{A}}(U) \equiv \langle \tilde{f}(U) \rangle_{\mathcal{A}}$, for $U \subseteq D$.*

Theorem 9. *Let D, D' be dioids, $f : D \rightarrow D'$ an $\mathcal{A}$ -morphism. Then $f_{\mathcal{A}} : \mathcal{A}[D] \rightarrow \mathcal{A}[D']$ is an identity-preserving quantale homomorphism; or, equivalently, a $\mathcal{P}$ -morphism.*

Proof. The identity $\langle 1 \rangle = \langle \{1\} \rangle$ is clearly preserved, since $f_{\mathcal{A}}(\langle 1 \rangle) = \langle \{f(1)\} \rangle = \langle 1 \rangle$. Products are preserved, since

$$f_{\mathcal{A}}(U \cdot V) = f_{\mathcal{A}}(\langle UV \rangle) = \langle \tilde{f}(UV) \rangle = \langle \tilde{f}(U)\tilde{f}(V) \rangle$$

while

$$f_{\mathcal{A}}(U) \cdot f_{\mathcal{A}}(V) = \langle \tilde{f}(U) \rangle \cdot \langle \tilde{f}(V) \rangle = \langle \tilde{f}(U)\tilde{f}(V) \rangle,$$

for $U, V \subseteq D$. Finally, suppose $\Upsilon \subseteq \mathcal{PD}$. Then

$$f_{\mathcal{A}}\left(\sum \Upsilon\right) = f_{\mathcal{A}}\left(\langle \bigcup \Upsilon \rangle\right) = \langle \tilde{f}(\bigcup \Upsilon) \rangle = \left\langle \bigcup_{U \in \Upsilon} \tilde{f}(U) \right\rangle,$$

while

$$\sum_{U \in \mathbf{Y}} f_{\mathcal{A}}(U) = \sum_{U \in \mathbf{Y}} \langle \tilde{f}(U) \rangle = \left\langle \bigcup_{U \in \mathbf{Y}} \tilde{f}(U) \right\rangle,$$

which establishes our result. In particular, the Morphism Lemma (lemma 6) is made use of in the second equality of each reduction to remove the inner bracket.

Free Quantale Extensions. This is the final ingredient needed to show that $\mathbf{Q}_{\mathcal{A}} : \mathbf{DA} \rightarrow \mathbf{DP}$ is a functor. Moreover, we may also show that the extension provided by the function is a free extension, in the sense of satisfying an appropriate universal property.

A functor must preserve identity morphisms. This is almost immediate. In fact, letting D be an $\mathcal{A}$ -dioid, then for the identity morphism $1_D : D \rightarrow D$, we have for $U \subseteq D$, $(1_D)_{\mathcal{A}}(U) = \langle \widetilde{1_D}(U) \rangle_{\mathcal{A}} = \langle U \rangle_{\mathcal{A}}$. Restricted to $U \in \mathcal{A}[D]$, this produces the result $(1_D)_{\mathcal{A}}(U) = \langle U \rangle_{\mathcal{A}} = U$. The preservation of the functor under composition is given by the following result.

Theorem 10. *Let D, D', D'' be dioids with $f : D' \rightarrow D''$ and $g : D \rightarrow D'$ being $\mathcal{A}$ -morphisms. Then $(f \circ g)_{\mathcal{A}} = f_{\mathcal{A}} \circ g_{\mathcal{A}}$.*

Proof. Let $U \subseteq D$. Then

$$f_{\mathcal{A}} \circ g_{\mathcal{A}}(U) = f_{\mathcal{A}}(\langle \tilde{g}(U) \rangle) = \langle \tilde{f}(\langle \tilde{g}(U) \rangle) \rangle = \langle \tilde{f}(\tilde{g}(U)) \rangle.$$

Reducing the left-hand side, we get

$$(f \circ g)_{\mathcal{A}}(U) = \langle (\widetilde{f \circ g})(U) \rangle = \langle \tilde{f}(\tilde{g}(U)) \rangle.$$

Thus, we finally arrive at the result

Corollary 5. *Let $\mathbf{Q}_{\mathcal{A}} : \mathbf{DA} \rightarrow \mathbf{DP}$ be given by $\mathbf{Q}_{\mathcal{A}}D \equiv \mathcal{A}[D]$, for $\mathcal{A}$ -dioids D , and $\mathbf{Q}_{\mathcal{A}}f \equiv f_{\mathcal{A}}$, for $\mathcal{A}$ -morphisms $f : D \rightarrow D'$ between $\mathcal{A}$ -dioids D and D' . Then $\mathbf{Q}_{\mathcal{A}}$ is a functor.*

The universal property is stated as follows. Letting Q denote a quantale with identity, we may define $\mathbf{Q}^{\mathcal{A}}Q$ as the algebra Q , itself, with only the $\mathcal{A}$ -dioid structure. This map is actually a functor $\mathbf{Q}^{\mathcal{A}} : \mathbf{DP} \rightarrow \mathbf{DA}$ which is termed a *forgetful functor*. It is nothing more than the identity map, where the extra structure associated with a $\mathcal{P}$ -dioid, not already present as part of the $\mathcal{A}$ -dioid structure, is forgotten.

The universal property states that any $\mathcal{A}$ -morphism $f : D \rightarrow \mathbf{Q}^{\mathcal{A}}Q$ from an $\mathcal{A}$ -dioid D should extend uniquely to a unit-preserving quantale morphism (or $\mathcal{P}$ -morphism) $f^* : \mathcal{A}[D] \rightarrow Q$. The sense in which this is an extension is that it works in conjunction with the unit $\mathcal{A}$ -morphism $\eta_D : D \rightarrow \mathcal{A}[D]$ defined by $\eta_D(d) = \langle d \rangle$, with $f(d) = f^*(\langle d \rangle)$. The functor pair $(\mathbf{Q}_{\mathcal{A}}, \mathbf{Q}^{\mathcal{A}})$ comprises an adjunction between $\mathbf{DA}$ and $\mathbf{DP}$ with a unit $D \mapsto \eta_D$. We will not directly prove this result here, since it will be superseded by the more general result in the following section.

3 A Hierarchy of Adjunctions

If we restrict the family of $\mathcal{A}$ -ideals to those generated by $\mathcal{B}$ -subsets, then we may obtain a representation for a $\mathcal{B}$ -algebra. Therefore, let us define the following:

Definition 6. Let D be a dioid, and $\mathcal{A}, \mathcal{B}$ be monadic operators. Then define $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D = \{\langle U \rangle_{\mathcal{A}} : U \in \mathcal{B}D\}$.

This is a generalization of our previous construction, with $\mathcal{A}[D] = \mathbf{Q}_{\mathcal{A}}^{\mathcal{P}}D$; or, $\mathbf{Q}_{\mathcal{A}} = \mathbf{Q}_{\mathcal{A}}^{\mathcal{P}}$. The algebra $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is closed under products. For, if $U, V \in \mathcal{B}D$, then $\langle U \rangle \cdot \langle V \rangle = \langle UV \rangle \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$, since $UV \in \mathcal{B}D$, by $\mathbf{A}_2$. Similarly, $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is also closed under sums from $\mathcal{B}\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$. Let $Z \in \mathcal{B}\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$. Since $U \in \mathcal{B}D \mapsto \langle U \rangle_{\mathcal{A}} \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is a monoid homomorphism, then by $\mathbf{A}_6$ it follows that $Z = \{\langle U \rangle_{\mathcal{A}} : U \in Y\}$ for some $Y \in \mathcal{B}\mathcal{B}D$. But, then we can write

$$\sum Z = \sum_{U \in Y} \langle U \rangle_{\mathcal{A}} = \left\langle \bigcup Y \right\rangle_{\mathcal{A}} \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D,$$

since, by $\mathbf{A}_3$, $\bigcup Y \in \mathcal{B}D$. Together, this proves the following result:

Theorem 11. Let D be a dioid and $\mathcal{A}, \mathcal{B}$ be monadic operators. Then $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is a $\mathcal{B}$ -dioid.

We also have closure under the lifting of $\mathcal{A}$ -morphisms:

Theorem 12. Let $\mathcal{A}$ and $\mathcal{B}$ be monadic operators. If D and D' are dioids and $f : D \rightarrow D'$ is an $\mathcal{A}$ -morphism, then $I \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D \rightarrow f_{\mathcal{A}}(I) \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D'$.

Proof. Let $I = \langle U \rangle$, with $U \in \mathcal{B}D$. Then $\tilde{f}(U) \in \mathcal{B}D'$, by $\mathbf{A}_4$. Therefore $f_{\mathcal{A}}(I) = \left\langle \tilde{f}(U) \right\rangle_{\mathcal{A}} \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D'$.

This allows us to generalize our previous result to the following:

Theorem 13. Let $\mathcal{A}$ and $\mathcal{B}$ be monadic operators. Define $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} : \mathbf{D}\mathcal{A} \rightarrow \mathbf{D}\mathcal{B}$ by: $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D = \{\langle U \rangle_{\mathcal{A}} : U \in \mathcal{B}D\}$ for $\mathcal{A}$ -dioids D , as before; $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}f = f_{\mathcal{A}}$ for $\mathcal{A}$ -morphisms $f : D \rightarrow D'$. Then $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}$ is a functor.

Theorem 14. Let $\mathcal{A}$ and $\mathcal{B}$ be monadic operators with $\mathcal{A} \geq \mathcal{B}$. Then $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} : \mathbf{D}\mathcal{A} \rightarrow \mathbf{D}\mathcal{B}$ is the forgetful functor. In particular, for $\mathcal{A} = \mathcal{B}$, $\mathbf{Q}_{\mathcal{A}}^{\mathcal{A}}$ is the identity functor on $\mathbf{D}\mathcal{A}$.

Proof. Under the stated condition, every ideal reduces to a principal ideal

$$U \in \mathcal{B}D \subseteq \mathcal{A}D \rightarrow \langle U \rangle_{\mathcal{A}} = \left\langle \sum U \right\rangle.$$

This establishes a one-to-one correspondence between $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ and D . Previously, we pointed out that the product is preserved with $\langle x \rangle \cdot \langle y \rangle = \langle xy \rangle$ for $x, y \in D$, and we already know that $\langle 1 \rangle = \langle \{1\} \rangle_{\mathcal{A}}$ is the identity. This shows that $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ and D are isomorphic as monoids.

Here, we can show that sums over $\mathcal{B}\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ exist in $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ without using property **A**₆ for $\mathcal{B}$. Suppose $Z \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$. Since the map $(\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}})^{-1} : \langle x \rangle \mapsto x$ is a monoid isomorphism then

$$V = (\widetilde{\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}})^{-1}(Z) = \{x \in D : \langle x \rangle \in Z\} \in \mathcal{B}D,$$

by **A**₄. Therefore,

$$\sum Z = \left\langle \sum_{v \in V} \langle v \rangle \right\rangle_{\mathcal{A}} = \left\langle \sum_{v \in V} \{v\} \right\rangle_{\mathcal{A}} = \langle V \rangle_{\mathcal{A}} \in \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D.$$

Therefore, $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is a $\mathcal{B}$ -dioid.

Thus, we only need to show that $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} : x \in D \mapsto \langle x \rangle$ is $\mathcal{B}$ -additive. To that end, let $U \in \mathcal{B}D$. Then, we have

$$\sum \widetilde{\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}}(U) = \sum_{u \in U} \langle u \rangle = \left\langle \bigcup_{u \in U} \{u\} \right\rangle_{\mathcal{A}} = \langle U \rangle_{\mathcal{A}} = \left\langle \sum U \right\rangle.$$

This shows that, as a $\mathcal{B}$ -dioid, $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}D$ is isomorphic to D .

Finally, we already know that $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}f = f_{\mathcal{A}}$ preserves arbitrary sums, for $\mathcal{A}$ -morphisms $f : D \rightarrow D'$. Therefore, $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}$ is a $\mathcal{B}$ -morphism. This establishes our result.

Finally, the following theorem shows the sense in which the hierarchy of monadic dioids may be considered as a chain of free extensions.

Theorem 15. *Let $\mathcal{A}$ and $\mathcal{B}$ be monadic operators with $\mathcal{A} \leq \mathcal{B}$. Then $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}$ is a left adjoint of $\mathbf{Q}_{\mathcal{B}}^{\mathcal{A}}$.*

Before proceeding with the proof, it will first be necessary to describe in more detail the result being sought out here. We are seeking to show that the functors $\mathbf{E} = \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}}$ and $\mathbf{U} = \mathbf{Q}_{\mathcal{B}}^{\mathcal{A}}$ forms an adjunction between the categories $\mathbf{D}\mathcal{A}$ and $\mathbf{D}\mathcal{B}$. This requires showing that there is a one-to-one correspondence between $\mathcal{A}$ -morphisms $f : A \rightarrow \mathbf{U}B$ and $\mathcal{B}$ -morphisms $g : \mathbf{E}A \rightarrow B$, for any $\mathcal{A}$ -dioid A and $\mathcal{B}$ -dioid B ; that is natural, in the sense that it respects compositions on both sides. Let the correspondence be denoted by the following rules

$$\frac{f : A \rightarrow \mathbf{U}B}{f^* : \mathbf{E}A \rightarrow B}, \frac{g : \mathbf{E}A \rightarrow B}{g_* : A \rightarrow \mathbf{U}B}.$$

To implement the one-to-one nature of the correspondence, we require

$$\frac{f : A \rightarrow \mathbf{U}B}{(f^*)_* = f}, \frac{g : \mathbf{E}A \rightarrow B}{(g_*)^* = g}.$$

To implement the *naturalness* condition, we require

$$\frac{g : A' \rightarrow A, f : A \rightarrow \mathbf{U}B, h : B \rightarrow B'}{(\mathbf{U}h \circ f \circ g)^* = h \circ f^* \circ \mathbf{E}g}.$$

The candidate chosen for this correspondence is $f^*(\langle U \rangle_{\mathcal{A}}) = \sum \sum \tilde{f}(U)$. But we must first show that this is well-defined. This is done through the following lemma, which is an elaboration of an argument presented originally in [8].

Lemma 7. *Let A be an $\mathcal{A}$ -doid and B a $\mathcal{B}$ -doid with $f : A \rightarrow \mathbf{UB}$ an $\mathcal{A}$ -morphism. For each $U \in \mathcal{B}A$, $\sum \tilde{f}(U) = \sum \tilde{f}(\langle U \rangle_{\mathcal{A}})$.*

Proof. It is important to note that this is also an existence result. Though $\tilde{f}(U) \in \mathcal{B}B$, by **A₄**, it need not be the case that $\tilde{f}(\langle U \rangle_{\mathcal{A}}) \in \mathcal{B}B$. Therefore, there is no guarantee at the outset that the latter be summable in B .

However, we do have the following result. Making use of the Morphism Lemma (lemma [6]), we know that

$$\langle \tilde{f}(U) \rangle_{\mathcal{A}} = \langle \tilde{f}(\langle U \rangle_{\mathcal{A}}) \rangle_{\mathcal{A}}$$

for any $U \in \mathcal{B}A$. Moreover, since $\tilde{f}(U) \in \mathcal{B}B$, by **A₄**, then the sum $\sum \tilde{f}(U) \in \mathcal{B}B$ is defined, and we can write

$$\langle \tilde{f}(\langle U \rangle_{\mathcal{A}}) \rangle_{\mathcal{A}} = \langle \tilde{f}(U) \rangle_{\mathcal{A}} = \langle \sum \tilde{f}(U) \rangle_{\mathcal{A}}.$$

This shows that $\sum \tilde{f}(U)$ is an upper bound of $\tilde{f}(\langle U \rangle_{\mathcal{A}})$. But it is already the least upper bound of the smaller set $\tilde{f}(U)$. Therefore, it must be the least upper bound of the larger set, as well.

On the basis of this result, the map $f^* : \mathbf{EA} \rightarrow B$ is well defined. With this matter resolved, we can then proceed to the proof of Theorem [15].

Proof (of Theorem [15]). That fact that $f \mapsto f^*$ is one-to-one comes from showing that f is recovered from the principal ideals by $f(x) = \langle f^*(\langle x \rangle) \rangle$. In particular, since $\langle x \rangle$ is an interval, then $\sum \tilde{f}(\langle x \rangle) = \langle \sum f(x) \rangle = f(x)$. Therefore,

$$\langle f^*(\langle x \rangle) \rangle = \langle \sum \sum \tilde{f}(\langle x \rangle) \rangle = \langle \sum f(x) \rangle = f(x).$$

To show that $f^* : \mathbf{EA} \rightarrow B$ is actually a $\mathcal{B}$ -morphism, we must first show that the monoid structure is preserved. For the identity, noting that $f(1) = \langle 1 \rangle \in \mathbf{UB}$, we have:

$$f^*(\langle 1 \rangle) = \sum \sum \tilde{f}(\{1\}) = \sum \sum \{f(1)\} = \sum f(1) = \sum \langle 1 \rangle = 1.$$

For products, we can write

$$\sum \sum \tilde{f}(UV) = \sum \sum \tilde{f}(U) \tilde{f}(V) = \sum \left(\sum \tilde{f}(U) \sum \tilde{f}(V) \right).$$

Noting that the sum on the right distributes and applying the definition of f^* , we obtain the result

$$f^*(\langle U \rangle_{\mathcal{A}} \cdot \langle V \rangle_{\mathcal{A}}) = f^*(\langle U \rangle_{\mathcal{A}}) f^*(\langle V \rangle_{\mathcal{A}}).$$

Next, we must show that the summation operator is preserved over $\mathcal{BEA}$. Let $Z \in \mathcal{BEA} = \mathcal{BQ}_{\mathcal{A}}^{\mathcal{B}}A$. It's at this point that we use property $\mathbf{A}_6$. Since $U \in \mathcal{BA} \mapsto \langle U \rangle_{\mathcal{A}} \in \mathcal{Q}_{\mathcal{A}}^{\mathcal{B}}A$ is a monoid homomorphism, then we may assume that there is a set $Y \in \mathcal{BBA}$ such that $Z = \{\langle U \rangle_{\mathcal{A}} : U \in Y\}$. Then the summation $f^*(\sum Z) = \sum \sum \tilde{f}(\langle U \rangle_{\mathcal{A}})$ can be rewritten, using the Sum Lemma (lemma 4), with

$$\langle \bigcup Z \rangle_{\mathcal{A}} = \left\langle \bigcup_{U \in Y} \langle U \rangle_{\mathcal{A}} \right\rangle_{\mathcal{A}} = \left\langle \bigcup_{U \in Y} U \right\rangle_{\mathcal{A}} = \langle \bigcup Y \rangle_{\mathcal{A}}.$$

Using the Morphism Lemma (lemma 6), we then have

$$\sum \tilde{f}(\langle \bigcup Z \rangle_{\mathcal{A}}) = \sum \tilde{f}(\langle \bigcup Y \rangle_{\mathcal{A}}) = \sum \tilde{f}(\bigcup Y).$$

The application to the union can be broken down to that on the component sets,

$$\sum \tilde{f}(\bigcup Y) = \sum \bigcup_{U \in Y} \tilde{f}(U).$$

Since each set $\tilde{f}(U) \in \mathcal{BB}$ (by property $\mathbf{A}_4$), the least upper bound $\sum \tilde{f}(U) \in \mathcal{BB}$ is defined. The associativity of least upper bounds, which is a general property of partially ordered sets, can then be used to write – making use, again, of the Sum Lemma –

$$\sum \bigcup_{U \in Y} \tilde{f}(U) = \sum \sum_{U \in Y} \tilde{f}(U) = \sum \sum_{U \in Y} \tilde{f}(\langle U \rangle_{\mathcal{A}}).$$

Similarly, applying associativity again, we can write

$$f^*(\sum Z) = \sum \sum \sum \tilde{f}(\langle U \rangle_{\mathcal{A}}) = \sum \sum \sum \tilde{f}(\langle U \rangle_{\mathcal{A}}).$$

From the other direction, we may write,

$$\sum \widetilde{f^*}(Z) = \sum_{U \in Y} f^*(\langle U \rangle_{\mathcal{A}}) = \sum_{U \in Y} \sum \sum \tilde{f}(\langle U \rangle_{\mathcal{A}}),$$

which establishes preservation of sums over $\mathcal{BEA}$.

The additional property of naturalness requires showing that this correspondence be well-behaved with respect to composition with morphisms from the respective categories. In particular, for an $\mathcal{A}$ -morphism $g : A' \rightarrow A$ and a $\mathcal{B}$ -morphism $h : B \rightarrow B'$, we need to show that $(\mathbf{U}h \circ f \circ g)^* = h \circ f^* \circ \mathbf{E}g$.

To this end, let $U \in \mathcal{BA}'$ and let I denote the interval $\sum \tilde{f}(\tilde{g}(\langle U \rangle_{\mathcal{A}})) \in \mathbf{UB}$. Noting, by the Morphism Lemma that $I = \sum \tilde{f}(\langle \tilde{g}(U) \rangle_{\mathcal{A}})$, we can write

$$(h \circ f^* \circ \mathbf{E}g)(\langle U \rangle_{\mathcal{A}}) = h(f^*(\mathbf{E}g(\langle U \rangle_{\mathcal{A}}))) = h\left(\sum I\right)$$

while

$$(\mathbf{U}h \circ f \circ g)^*(\langle U \rangle_{\mathcal{A}}) = \sum \sum \widetilde{\mathbf{U}h}(\tilde{f}(\tilde{g}(\langle U \rangle_{\mathcal{A}}))) = \sum \mathbf{U}h(I).$$

Since I is an interval in B , then $\sum Uh(I) = h(\sum I)$ follows, which establishes the result.

It is worth pointing out that $\mathbf{E}UB \cong B$. The ideal $\langle U \rangle_{\mathcal{A}} = \langle \sum U \rangle$ is principal, noting that $\sum U \in B$ is defined for all $U \in \mathcal{B}B$, since B is a $\mathcal{B}$ -dioid. The map $g_{\mathcal{A}}$ applied to this ideal results in

$$g_{\mathcal{A}}(\langle U \rangle_{\mathcal{A}}) = \langle \tilde{g}(\langle U \rangle_{\mathcal{A}}) \rangle_{\mathcal{A}} = \langle \tilde{g}(U) \rangle_{\mathcal{A}} = \left\langle \sum \tilde{g}(U) \right\rangle = \left\langle g\left(\sum U\right) \right\rangle$$

for a $\mathcal{B}$ -morphism $g : B \rightarrow B'$. Therefore, the composition $\mathbf{E} \circ \mathbf{U}$ is just the identity functor on $\mathbf{DB}$.

Corollary 6. *Let $\mathcal{A}, \mathcal{B}$ be monadic operators with $\mathcal{A} \leq \mathcal{B}$. Then $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} \circ \mathbf{Q}_{\mathcal{B}}^{\mathcal{A}}$ is the identity functor on $\mathbf{DB}$.*

In addition, we may show that the adjunctions behave consistently under compositions.

Corollary 7. *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be monadic operators with $\mathcal{A} \leq \mathcal{B} \leq \mathcal{C}$. Then $\mathbf{Q}_{\mathcal{V}}^{\mathcal{U}} \circ \mathbf{Q}_{\mathcal{W}}^{\mathcal{V}} = \mathbf{Q}_{\mathcal{W}}^{\mathcal{U}}$, for any permutation $\mathcal{U}, \mathcal{V}, \mathcal{W}$ of $\mathcal{A}, \mathcal{B}, \mathcal{C}$.*

Proof. It is actually only necessary to take $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ or $(\mathcal{C}, \mathcal{B}, \mathcal{A})$ as the permutations of $(\mathcal{U}, \mathcal{V}, \mathcal{W})$ since the other cases can be derived by composition using corollary 6.

These two cases result from showing that adjunctions are closed under composition which is a general category-theoretic result. The adjunctions here involve left-adjoints of forgetful functors. However, since the forgetful functors close under composition, and the composition of adjunctions is also an adjunction, then the result follows directly from the uniqueness of left adjoints [10] (Corollary 1, p. 83).

Theorem 16. *The functor $\mathcal{A} : \mathbf{Monoid} \rightarrow \mathbf{DA}$ and the forgetful functor $\hat{\mathcal{A}} : \mathbf{DA} \rightarrow \mathbf{Monoid}$ form an adjunction pair.*

Proof. This is the essence of the properties **A**₁-**A**₄. Here, the unit $\eta_M : M \rightarrow \mathcal{A}M$ is the inclusion $\eta_M(m) = \{m\}$. The extension of the monoid homomorphism $f : M \rightarrow \hat{\mathcal{A}}\mathcal{A}$ to an $\mathcal{A}$ -morphism $f^* : \mathcal{A}M \rightarrow \mathcal{A}$ is related to the least upper bound operator by $f^*(U) = \sum \tilde{f}(U)$, for $U \in \mathcal{A}M$. The naturalness of this correspondence is, in fact, the essential point of Theorem 11. In fact, the construction of $\mathcal{A}$ -dioids is a special case of a general construction, through adjunctions, of what are known in category theory as T-algebras [10]. To complete the proof will actually require establishing properties

$$\mathbf{D}_4 \quad \sum \{m\} = m \text{ for } m \in D,$$

$$\mathbf{D}_5 \quad \sum_{U \in Y} \sum U = \sum (\bigcup Y), \text{ for } Y \in \mathcal{A}AD,$$

$$\mathbf{D}_6 \quad f(A) = \sum_{a \in A} f(\{a\}), \text{ for } A \in \mathcal{A}M, \text{ where } f : \mathcal{A}M \rightarrow D \text{ is an } \mathcal{A}\text{-morphism}$$

which are all elementary consequences for partially ordered sets.

It follows, also, from these considerations that $\mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} \circ \mathcal{A} = \mathcal{B}$ for $\mathcal{A} \leq \mathcal{B}$ and that, under the same condition, $\hat{\mathcal{B}} \circ \mathbf{Q}_{\mathcal{A}}^{\mathcal{B}} = \hat{\mathcal{A}}$.

4 Further Developments

What we have done is construct a hierarchy of monads. For each operator $\mathcal{A}$ there is an adjunction pair $(\mathcal{A}, \hat{\mathcal{A}})$ that extends the category of monoids to the category of $\mathcal{A}$ -dioids. The unit of the adjunction is the polymorphic function (i.e., natural transformation) $\eta : I_{\mathbf{Monoid}} \rightarrow \hat{\mathcal{A}} \circ \mathcal{A}$, given by $\eta_M : M \rightarrow \mathcal{A}M$, where $\eta_M(m) = \{m\}$. The monad product $\Sigma : \mathcal{A} \circ \hat{\mathcal{A}} \rightarrow I_{\mathbf{DA}}$ is given by $\Sigma_D : \mathcal{A}D \rightarrow D$, where $\Sigma_D(U) = \sum U$.

The incorporation of the idempotency property, $A = A + A$, is the critical feature behind the occurrence of the partially ordered monoid structure. In contrast, in the formal power series approach [6,7,11], addition no longer need be idempotent. Therefore, a natural route of generalization is of the monad hierarchy from dioids to semirings. Unlike the case for dioids, where a Σ operator is already given to us satisfying all of $\mathbf{D}_1, \dots, \mathbf{D}_6$, for a semiring-based formulation of the foregoing the additional properties $\mathbf{D}_4, \mathbf{D}_5, \mathbf{D}_6$ will also need to be explicitly stipulated.

Acknowledgments. The author would like to thank Dexter Kozen and Bernhard Möller for their assistance, Bruce Litow for his encouragement and support, and Derick Wood for inspiring research in the area of algebraizing formal language and automata theory.

References

1. Hopkins, M.W.: The Algebraic Approach I: The Algebraization of the Chomsky Hierarchy. RelMiCS 2008 (to be published, 2008)
2. Gruska, J.: A Characterization of Context-Free Languages. Journal of Computer and System Sciences 5, 353–364 (1971)
3. McWhirtier, I.P.: Substitution Expressions. Journal of Computer and System Sciences 5, 629–637 (1971)
4. Yntema, M.K.: Cap Expressions for Context-Free Languages. Information and Control 8, 311–318 (1971)
5. Ésik, Z., Leiss, H.: Algebraically complete semirings and Greibach normal form. Ann. Pure. Appl. Logic. 133, 173–203 (2005)
6. Ésik, Z., Kuich, W.: Rationally Additive Semirings. Journal of Computer Science 8, 173–183 (2002)
7. Berstel, J., Reutenauer, C.: Les Séries Rationnelles et Leurs Langues. Masson (1984). English edition: Rational Series and Their Languages. Springer, Heidelberg (1988)
8. Kozen, D.: On Kleene Algebras and Closed Semirings. In: Rovan, B. (ed.) MFCS 1990. LNCS, vol. 452, pp. 26–47. Springer, Heidelberg (1990)

9. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall, London (1971)
10. MacLane, S.: Categories for the Working Mathematician. Springer, Heidelberg (1971)
11. Kuich, W., Salomaa, A.: Semirings, Automata and Languages. Springer, Berlin (1986)
12. Chomsky, N., Schützenberger, M.P.: The Algebraic Theory of Context-Free Languages. In: Braort, P., Hirschberg, D. (eds.) Computer Programming and Formal Systems, pp. 118–161. North-Holland, Amsterdam (1963)

Automated Reasoning for Hybrid Systems

— Two Case Studies —

Peter Höfner

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany
`hoefner@informatik.uni-augsburg.de`

Abstract. At an abstract level hybrid systems are related to variants of Kleene algebra. Since it has recently been shown that Kleene algebras and their variants, like omega algebras, provide a reasonable base for automated reasoning, the aim of the present paper is to show that automated algebraic reasoning for hybrid system is feasible. We mainly focus on applications. In particular, we present case studies and proof experiments to show how concrete properties of hybrid systems, like safety and liveness, can be algebraically characterised and how off-the-shelf automated theorem provers can be used to verify them.

1 Introduction

Hybrid systems are heterogeneous systems characterised by the interaction of discrete and continuous dynamics. Because of their widespread applications there was a rapid growth of interest in such systems during the last decade. They are an effective tool for modelling, design and analysis of a large number of technical systems such as traffic control [9,13] and automated manufacturing [8].

The most elementary and classical hybrid system usually consists of a controller and a controlled subsystem. Usually the controller represents discrete behaviour and the environment is described by the continuous behaviour. In general, the behaviour of the controller depends on the state and the behaviour of the controlled system and cannot be considered in isolation. More complicated hybrid systems usually arise by composing smaller systems.

Nearly from the beginning of their formal introduction in computer science it was proposed to model hybrid systems as hybrid automata [11,14]. Hybrid automata are based on timed automata [4] and have, next to nodes and edges, differential equations and variables. These additional features reflect the behaviour of the environment in each node. The study of hybrid systems in computer science is still largely focused on hybrid automata. There are only few other approaches to hybrid systems, e.g., [5]. In [17] an approach that combines variants of Kleene algebra with the concept of hybrid systems is given.

Over the last decades Kleene algebras have proved to be fundamental first-order structures in computer science with widespread applications ranging from program analysis and semantics to combinatorial optimisation and concurrency control. They offer operators for modelling actions, programs or state transitions under non-deterministic choice, sequential composition and finite iteration. They

allow the formalisation and specification of safety and liveness properties for hybrid systems at an abstract level.

Recently, it has been shown that Kleene algebra and their variants provide a reasonable base for automated deduction [20,21]. Therefore the techniques developed there should be reusable for automated reasoning about hybrid systems in an algebraic setting. The aim of the paper is to show that the algebraic approach indeed yields proofs for safety and liveness, and to discover if automated algebraic reasoning for hybrid system is feasible.

This paper mainly focuses on applications. In particular, we present case studies to show how properties can be algebraically specified and how off-the-shelf automated theorem provers can be used to verify them. The first case study is a technical system where a selected route is automatically compared with the specification. If the specification is not satisfied another route has to be chosen. This case study is developed step by step to briefly define and illustrate the underlying theory. The second case study is more complex and describes an assembly line scheduler.

2 Case Study I—Checking a Specification

To illustrate the basic definitions and concepts used in the remainder, we take the following example.

Example 2.1. We assume a security service that has to control three locations (bank, disco and university). The corresponding hybrid automaton (Figure 1) models all possible routes the security service can use when starting at university.

We briefly explain the meaning of the automaton. Details about hybrid automata in general can be found in [3,14]. Employees of the security service can

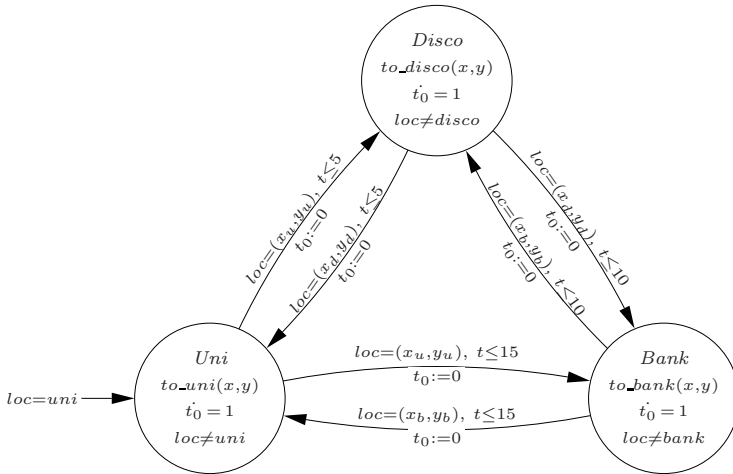


Fig. 1. A simple system for route planning

be in three different states: either they travel to university (described by state *Uni*) or they are on their way to the *Bank* or they are going to control the *Disco*. The functions to_uni and t_0 describe the continuous behaviour of the hybrid system when moving to university (continuous behaviour in node *Uni*): $to\_uni(t)$ determines the path to university starting from the actual time and the current position given by the two coordinates x_c and y_c . Usually this function is specified by an initial value problem combined with (ordinary) differential equations. To measure time between two locations a clock (the function t_0) is introduced. Special locations for university, bank and disco are denoted by (x_u, y_u) , (x_b, y_b) and (x_d, y_d) , respectively. As long as the university is not reached (denoted by the invariant condition $loc \neq (x_u, y_u)$), the security service continues to move towards the university. If the university is reached ($loc = (x_u, y_u)$), the employees have the (non-deterministic) choice to go either to the bank or to the disco. This state-changing situation represents the discrete part of the hybrid system. Typically, this decision is made by a controller. The other states and functions are built in a similar way. The time conditions like $t_0 \leq 5$, given at the edges, guarantee that the way between uni and disco takes at most 5 minutes; the way between disco and bank needs less than 10 minutes and the one between bank and uni less than 15 minutes. After changing the state the clock is reset to 0. Now we assume that the security service has to check every place at least every half an hour. Due to the small size it is easy to see that e.g. the circle starting at university and then via bank to disco and back to university satisfies the required safety condition, if it is repeated again and again.

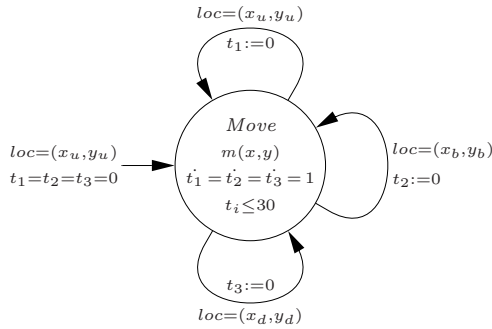


Fig. 2. An alternative route planning automaton

To encode the time constraint that every location has to be visited every 30 minutes, one can use the hybrid automaton of Figure 2. The main idea is to have one state in which the service is moving. The action of moving is denoted by $m(t)$, e.g., $\dot{m}(t) = v$ if the movement is done with a constant velocity v , and the current position as initial condition $m(0) = (x_c, y_c)$.¹ Unfortunately, in this automaton the time constraints between the 3 locations cannot be encoded. To

¹ This example is not realistic, but will illustrate the crucial ideas.

model the specification within hybrid automata one has to combine both automata presented. This yields an automaton with 4 clocks. To check the given safety property using one of these hybrid automata is not an easy and straightforward exercise. $\square$

But how can it be (automatically) checked that a given run of a hybrid automaton satisfies a given specification, in general? The example above shows that it is not easy to determine an answer. In the remainder we show that in an algebraic setting the above safety property yields a surprisingly simple inequality that can easily be proved.

3 An Algebra for Hybrid Systems

We aim at the use of first-order automated reasoning for hybrid systems. For that an algebraic (first-order) view of hybrid systems is needed. We follow the lines of [17]. The algebra for hybrid systems uses trajectories that reflect the variation of the values of the variables over time.

Let V be a set of *values* and D a set of *durations* (e.g. $\mathbb{N}, \mathbb{Q}, \mathbb{R}, \dots$). We assume that $(D, +, 0)$ is a commutative monoid and the relation $x \leq y \Leftrightarrow_{df} \exists z. x + z = y$ is a linear order on D . If $+$ is cancellative, 0 is the least element and $+$ is isotone w.r.t. $\leq$. Moreover, 0 is indivisible. D may include the special value ∞ . If so, ∞ is required to be an annihilator w.r.t. $+$ and hence the greatest element of D (and cancellativity of $+$ is restricted to elements in $D - \{\infty\}$). For $d \in D$ we define the interval $\text{intv } d$ of admissible times as

$$\text{intv } d =_{df} \begin{cases} [0, d] & \text{if } d \neq \infty \\ [0, d[& \text{otherwise} \end{cases}.$$

A *trajectory* t is a pair (d, g) , where $d \in D$ and $g : \text{intv } d \rightarrow V$. Then d is the *duration* of the trajectory, the image of $\text{intv } d$ under g is its *range* $\text{ran}(d, g)$. This view models *oblivious* systems in which the evolution of a trajectory is independent of the history before the starting time.

The idea of composing two trajectories $T_1 = (d_1, g_1)$ and $T_2 = (d_2, g_2)$ is to extend T_1 at the right end, i.e., at time d_1 , with T_2 to a trajectory $(d_1 + d_2, g)$, if reasonable. Figure 3 illustrates the concept. Since g needs to be a function, one needs to decide how to handle the time-point d_1 . The definition of sequential composition is given by

$$(d_1, g_1) \cdot (d_2, g_2) =_{df} \begin{cases} (d_1 + d_2, g) & \text{if } d_1 \neq \infty \wedge g_1(d_1) = g_2(0) \\ (d_1, g_1) & \text{if } d_1 = \infty \\ \text{undefined} & \text{otherwise} \end{cases}$$

with $g(x) = g_1(x)$ for all $x \in [0, d_1]$ and $g(x + d_1) = g_2(x)$ for all $x \in \text{intv } d_2$.

For a zero-length trajectory $(0, g_1)$ we have $(0, g_1) \cdot (d_2, g_2) = (d_2, g_2)$ if $g_1(0) = g_2(0)$. Similarly, $(d_2, g_2) \cdot (0, g_1) = (d_2, g_2)$ if $g_1(0) = g_2(d_2)$ or $d_2 = \infty$. For a value $v \in V$, let $\underline{v} =_{df} (0, g)$ with $g(0) = v$ be the corresponding zero-length trajectory.

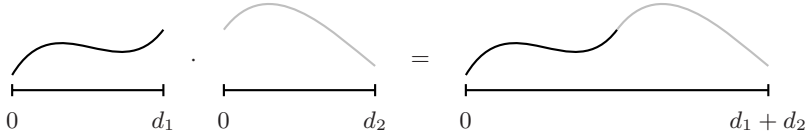


Fig. 3. Composition of two finite trajectories

A *process* is a set of trajectories, consisting of possible behaviours of a hybrid system. The set of all processes is denoted by PRO . The finite and infinite parts of a process A are defined as

$$\inf A =_{df} \{(d, g) \in A \mid d = \infty\} \quad \text{fin } A =_{df} A - \inf A$$

Composition is lifted to processes as follows:

$$A \cdot B =_{df} \inf A \cup \{a \cdot b \mid a \in \text{fin } A, b \in B\}.$$

The constraint $g_1(d_1) = g_2(0)$ for composability of trajectories $T_1 = (d_1, g_1)$ and $T_2 = (d_2, g_2)$ is very restrictive in a number of situations. Hence a *compatibility relation*, which describes the behaviour at the point of composition is introduced in [18]. That relation allows ‘jumps’ at the connection point between T_1 and T_2 . In the remainder we do not need this concept; we mention it only for completeness.

Example 3.1. We want to give an algebraic expression for the automaton of Figure 1. For that we define $V = \mathbb{R}^2$, where an element determines the current position (x, y) . A possible way is to define a process for each node for a hybrid automaton. For example

$$u =_{df} \{(d, g) \mid g(t) = \text{to\_uni}(t)\}.$$

The clock t_0 can be dropped since we have the duration d available and therefore the clock is redundant. Similar to u one can define processes for the nodes *Disco* and *Bank*. But, since the functions to_uni , to_bank and to_disco are not specified we abstract to a general “move action”. In particular, we define

$$a_n =_{df} \{(d, g) \mid d \leq n, g = m(t)\}.$$

It describes all routes that the security service can use and take at most n minutes. To check if the security service is at a certain point, we use zero-length trajectories:

$$\begin{aligned} at_u &=_{df} \underline{(x_u, y_u)} = \{(0, g) \mid g(0) = (x_u, y_u)\}, \\ at_b &=_{df} \underline{(x_b, y_b)} = \{(0, g) \mid g(0) = (x_b, y_b)\}, \\ at_d &=_{df} \underline{(x_d, y_d)} = \{(0, g) \mid g(0) = (x_d, y_d)\}. \end{aligned}$$

These sets describe the situation when the security service is exactly at the locations university (at_u), bank (at_b) and disco (at_d). In the remainder we use such elements to model tests and assertions. Now, we are able to describe the hybrid automaton of Figure 1 in an algebraic setting. The main construct is of the form $at_u \cdot a_5 \cdot at_d$ which describes all possible ways from university to the disco taking at most 5 minutes. The whole automaton can be described by

$$at_u \cdot \left(at_u \cdot a_5 \cdot at_d \cup at_d \cdot a_5 \cdot at_u \cup \right. \\ \left. at_d \cdot a_{10} \cdot at_b \cup at_b \cdot a_{10} \cdot at_d \cup \right. \\ \left. at_b \cdot a_{15} \cdot at_u \cup at_u \cdot a_{15} \cdot at_b \right)^\omega, \quad (1)$$

where $^\omega$ models infinite iteration and therefore an infinite loop. The exact definition of this iteration operator is given in the next section. $\square$

4 Algebraic Background

Let us have a closer look at the algebraic structure of the trajectory-based model.

A *left semiring* is a quintuple $(S, +, 0, \cdot, 1)$ where $(S, +, 0)$ is a commutative monoid and $(S, \cdot, 1)$ is a monoid such that $\cdot$ is left-distributive over $+$ and *left-strict*, i.e., $0 \cdot a = 0$. The left semiring is *idempotent* if $+$ is idempotent and $\cdot$ is right-isotone, i.e., $b \leq c \Rightarrow a \cdot b \leq a \cdot c$, where the *natural order* $\leq$ on S is given by $a \leq b \Leftrightarrow_{df} a + b = b$. Left-isotony of $\cdot$ follows from its left-distributivity. Moreover, 0 is the $\leq$ -least element. A *semiring* is a left semiring in which $\cdot$ is also right-distributive and right-strict. The latter axiom (right-strictness) is dropped to model infinite behaviour. Differences between left semirings and standard semirings are listed e.g. in [25].

An idempotent left semiring S is called a *left quantale* if S is a complete lattice under the natural order and $\cdot$ is universally disjunctive in its left argument. Following [7], one might also call a left quantale a *left standard Kleene algebra*. A left quantale is *Boolean* if its underlying lattice is a Boolean algebra. In these cases the meet operator $\sqcap$ is available, too.

By simple calculations we get the two splitting laws

$$a + b \leq c \Leftrightarrow a \leq c \wedge b \leq c \quad \text{and} \quad a \leq b \sqcap c \Leftrightarrow a \leq b \wedge a \leq c. \quad (2)$$

An important left semiring (that is even a semiring and a left quantale) is REL, the algebra of binary relations over a set under relational composition.

Checking all the axioms for the case of processes, we get

Lemma 4.1

1. The processes form a Boolean left quantale $\text{PRO} =_{df} (\mathcal{P}(\text{TRA}), \cup, \emptyset, \cdot, I)$ with $I =_{df} \{(0, g) \mid (0, g) \in \text{TRA}\}$.
2. Additionally, $\cdot$ is positively disjunctive in its right argument.

A *left Kleene algebra* is a structure $(S, *)$ consisting of an idempotent semiring S and an operation $*$ that satisfies the left *unfold* and *induction* axioms

$$1 + a \cdot a^* \leq a^*, \quad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c.$$

Informally, the $*$ -operator characterises finite iteration. To express infinite iteration we axiomatise an ω -operator over a left Kleene algebra. A *left omega algebra* [25] is a pair (S, ω) such that S is a left Kleene algebra and ω satisfies the *unfold* and *coinduction* axioms

$$a^\omega = a \cdot a^\omega, \quad c \leq a \cdot c + b \Rightarrow c \leq a^\omega + a^* \cdot b.$$

As a consequence of fixpoint fusion (e.g. [10]) we have the following lemma.

Lemma 4.2. 1. *Every left quantale can be extended to a left Kleene algebra by defining $a^* =_{df} \mu x. a \cdot x + 1$.*
 2. *If the left quantale is a completely distributive lattice then it can be extended to a left omega algebra by setting $a^\omega =_{df} \nu x. a \cdot x$. In this case,*

$$\nu x. a \cdot x + b = a^\omega + a^* \cdot b.$$

The following lemma lists a couple of properties for left omega algebras which are needed afterwards. Some of them can be found in [25].

Lemma 4.3. *Assume a left omega algebra S and $a, b \in S$.*

1. $a \cdot (b \cdot a)^\omega \leq (a \cdot b)^\omega$.
2. $a^\omega \cdot b \leq a^\omega$.
3. $(a \cdot b)^\omega \leq (a + b)^\omega$.
4. $\forall i \in \mathbb{N}, i > 0 : (a^i)^\omega \leq (a^+)^\omega = a^\omega$, where $a^+ =_{df} a^* \cdot a$.

All proofs (except the first inequality of Lemma 4.3[4]) have been done by the automated theorem prover Prover9 (cf. Section 5) and can be found at a website [19]. The property $(a^i)^\omega \leq (a^+)^\omega$ cannot be encoded with Prover 9 because it is universally quantified. But it is a simple consequence of $a^i \leq a^+$ and isotony.

In Example 3.1, we have already used sets of zero-length trajectories to model assertions. The algebraic counterparts of such elements are tests in (left) semirings (e.g. [12,23]). One defines a *test* in an idempotent left semiring (quantale) to be an element $p \leq 1$ that has a complement q relative to 1, i.e., $p + q = 1$ and $p \cdot q = 0 = q \cdot p$. The set of all tests of S is denoted by $\text{test}(S)$. It is not hard to show that $\text{test}(S)$ is closed under $+$ and $\cdot$ and has 0 and 1 as its least and greatest elements. Moreover, the complement $\neg p$ of a test p is uniquely determined by the definition and $\text{test}(S)$ forms a Boolean algebra. In particular, tests are idempotent w.r.t. multiplication and we have the *shunting rule* for a test p :

$$p \cdot (p \cdot a)^\omega = (p \cdot a)^\omega \quad \text{and} \quad (p \cdot a)^\omega = (p \cdot a \cdot p)^\omega. \quad (3)$$

Again, the proofs can be done fully automatically using Prover9 (see Section 5). Due to Lemma 4.1 and 4.2, we also have finite iteration $*$ and infinite iteration ω with all their laws available in PRO. Moreover we can now formulate the specification of Example 2.1

Example 4.4. Remember that we want to check that, for a given trajectory of the hybrid automaton, the security service checks every location at least every 30 minutes. Let us consider the following (infinite) route for the security service.

$$\tau =_{df} (at_u \cdot a_5 \cdot at_d \cdot a_{10} \cdot at_b \cdot a_{15})^\omega .$$

It is straightforward to show that τ is a trace of the hybrid automaton's encoding of Figure 1 (cf. Equation (1)). To formulate the safety criterion for visiting each place at least once in 30 minutes, we have to check $\tau \leq (a_{30} \cdot at_u)^\omega \sqcap (a_{30} \cdot at_d)^\omega \sqcap (a_{30} \cdot at_b)^\omega$. By (2) it is equivalent that

$$\tau \leq (a_{30} \cdot at_u)^\omega , \quad \tau \leq (a_{30} \cdot at_d)^\omega \quad \text{and} \quad \tau \leq (a_{30} \cdot at_b)^\omega . \quad (4)$$

We only show that the second equation can easily be checked by hand; the other inequalities can be shown similarly. In the next section we present a possibility to automate such calculations. By isotony and definition of a_n we get

$$at_u \cdot a_5 \cdot at_d \cdot a_{10} \cdot at_b \cdot a_{15} \leq a_5 \cdot at_d \cdot a_{10} \cdot a_{15} \leq a_5 \cdot at_d \cdot a_{25} .$$

Therefore it is sufficient to show that $(a_5 \cdot at_d \cdot a_{25})^\omega \leq (a_{30} \cdot at_d)^\omega$. By unfold, Lemma 4.3, isotony, and unfold:

$$\begin{aligned} & (a_5 \cdot at_d \cdot a_{25})^\omega \\ &= (a_5 \cdot at_d \cdot a_{25}) \cdot (a_5 \cdot at_d \cdot a_{25})^\omega \\ &\leq a_5 \cdot at_d \cdot (a_{25} \cdot a_5 \cdot at_d)^\omega \\ &\leq a_{30} \cdot at_d \cdot (a_{30} \cdot at_d)^\omega \\ &= (a_{30} \cdot at_d)^\omega . \end{aligned}$$

This calculation shows that the chosen trace satisfies the safety criterion. In the algebraic setting it is a simple and short calculation, whereas in the setting of hybrid automata it was not possible in a straightforward way. $\square$

5 Automated Deduction

Having the algebraic characterisation of hybrid systems we can now use off-the-shelf theorem provers to verify or falsify properties. We use McCune's Prover9 tool [24] for proving theorems, but any first-order theorem prover should lead to similar results. Kleene algebras have already been integrated into higher-order theorem provers [1, 22, 29] and their applicability as a formal method has successfully been demonstrated in that setting. Nevertheless higher-order theorem provers need a huge amount of user interaction, whereas first-order provers need no interaction at all.

Prover9 is a saturation-based theorem prover for first-order equational logic. It implements an ordered resolution and paramodulation calculus and, by its treatment of equality by rewriting rules and Knuth-Bendix completion, it is particularly suitable for reasoning within variants of semirings. Prover9 is complemented by the counterexample generator Mace4, which is very useful in practice.

Prover9 and Mace4 accept input in a syntax for first-order equational logic. The input file consists essentially of a list of hypotheses (the set of support), e.g., the axioms of left omega algebra, and a goal to be proved. Prover9 negates the goal, transforms the hypotheses and the goal into clausal normal form and tries to produce a refutation. Mace4, in contrast, enumerates finite models of the hypothesis and checks whether they are consistent with the goal.

The inference process of saturation-based theorem proving is discussed in detail in the Handbook on Automated Reasoning [28]. Roughly, it consists of two interleaved modes.

- The deduction mode closes a given clause set under the inference rules of resolution, factoring and paramodulation. The paramodulation rule implements equational reasoning by replacing equals by equals.
- The simplification mode discards clauses from the working set if they are redundant with respect to other clauses.

In this process, simplification rules are applied eagerly and deduction rules lazily to keep the working set small. The process stops when the closure has been computed or when the empty clause $\$F$ — which denotes inconsistency — has been produced. Obviously the termination cannot be guaranteed. In the second case, Prover9 reconstructs and displays a proof.

Saturation-based theorem proving implements a semi-decision procedure for first-order equational logic. Whenever the goal is entailed by the hypotheses, the empty clause can be produced in finitely many steps. Otherwise, if the goal is not entailed, a counterexample exists, though not necessarily a finite one.

Since we are interested in robust results that can quickly be obtained by non-experts, we use the prover more or less as a black box and rely on the default strategies provided by Prover9. This makes our experiments more relevant to formal software development contexts.

First we have to encode left omega algebra for Prover9. This is done in a straightforward way; the code can be found in Appendix B. The goal to be proved is also encoded in the same way, i.e., to prove Lemma 4.3.11 one has to add the lines

```
formulas(goals).
  x;(y;x)^ + (x;y)^ = (x;y)^.
end_of_list.
```

whereas $;$ denotes multiplication, $+$ denotes addition and $\wedge$ denotes the omega operator. The proof takes around 100s and is fully automatically.² To speed up the proofs one can use hypotheses learning techniques [21,30]. This reduces the set of axioms and yields a proof in less than a second for the above equation. Such techniques seem very promising since the simple first-order equational calculus of idempotent left semirings (left Kleene algebras/left omega algebra) yields particularly short proofs. Let us now return to our running example.

² We use a Pentium 4, 3 GHz with Hyper-Threading, 2 GB RAM.

Example 5.1. We will now check the Equations (4) fully automatically. Since standard theorem provers are not able to handle simple arithmetics, we have to encode the relationship between different elements like $a_5 \cdot a_{15} \leq a_{30}$ by hand. But, obviously it is not difficult to produce such formulas with an automated preprocessor. The three equations are encoded by

```

formulas(goals).
  all all u all d all b(
    u;u=u & u+1=1 & d;d=d & d+1=1 & b;b=b & b+1=1      %preconditions
  ->
    (u;a5;d;a10;b;a15)^ ~ (a30;u)^ ~ (a30;u)^ &
    (u;a5;d;a10;b;a15)^ ~ (a30;d)^ ~ (a30;d)^ &      %the 3 equations
    (u;a5;d;a10;b;a15)^ ~ (a30;b)^ ~ (a30;b)^).
end_of_list.

```

In the code u corresponds to at_u , d to at_d , $a5$ to a_5 , etc. Since at_u , at_d and at_b are zero-length processes and therefore tests, we have to specify tests for Prover9. This can be done in a general setting (see [19]) or by specifying properties of tests. The preconditions reflect the two main properties for tests, namely that tests are idempotent and subidentities. Prover9 shows each of the equations in about 5 s. Their conjunction takes several minutes. The full input and output files as well as further information including the number of proofsteps and exact running times, can be found at [19]. The files also show how the needed arithmetic is encoded. $\square$

So far we have shown that algebraic reasoning for hybrid systems is feasible. In particular, we have presented a safety property for a concrete hybrid system. Furthermore we have encoded the property with the off-the-shelf theorem prover Prover9 and have proved it fully automatically. Therefore our algebra provides an interesting new way of verifying hybrid systems. Other approaches are discussed in Section 7. It is straightforward to extend the above example. For instance, one can add more locations or one can refine the safety property (e.g., “The security service has to drive to a petrol station every 10 hours and refuel there for 5 minutes”.) All these extensions do not change the algebra and/or the way of verifying the specification.

Verifying larger systems might need more time to prove properties fully automatically. But, checking properties are usually done in advance and not in real time. Moreover Prover9 can prove even complex properties in reasonable time; see e.g. Back’s atomicity refinement law in [21]. Therefore we expect that one can use our approach for larger systems, too.

6 Case Study II—An Assembly Line Scheduler

To further underpin our approach we sketch a more complex example: an assembly line scheduler that must assign elements from an incoming stream to one of two assembly lines [15].

New parts occur every four minutes in the stream. The lines themselves process the parts at different speeds: jobs travel between one and two meters per minute on the first line, while on the second the speed is between two and three

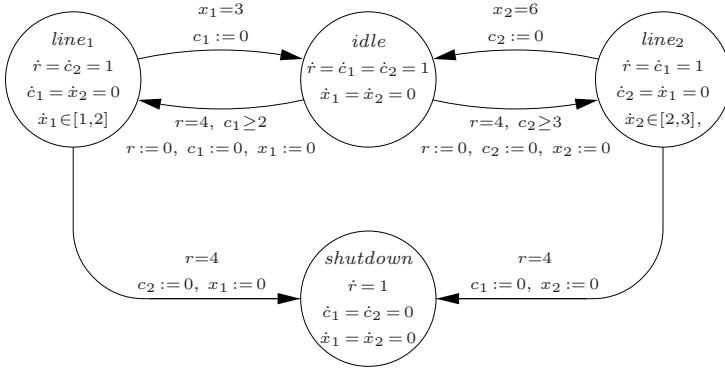


Fig. 4. Two assembly lines

metres per minute. The first line is three metres, the second six metres long. Once the lines finish a job, they insert cleaning phases of two and three minutes, respectively, during which no job can be taken up. The whole system accepts a job if both lines are free, and at most one is cleaning up. If the system cannot accept a job it shuts down.

The system is modeled by a hybrid automaton (Figure 4). There are four states: in *idle* no jobs are being processed; in *line₁* and *line₂* the lines for processing jobs are modelled; in *shutdown* the system shuts down. The variables x_1 and x_2 measure the distance a job has travelled along the first and second line, respectively. The variable c_1 and c_2 indicate the amount of time for cleaning up. Finally the variable r measures the elapsed time since the last arrival of a job.

As a liveness property one wants to avoid the system to go down. In [16] it is mentioned that any feasible schedule must choose the first line infinitely often. We will characterise this liveness property in our algebraic setting. Similar to Section 3 we define sets of trajectories l_1 , l_2 , i and s for the nodes *line₁*, *line₂*, *idle* and *shutdown* respectively (see Appendix A for the definitions). Since s is an error state we further assume that the corresponding process only consists of trajectories of infinite length. (If it is reached once, it will never be left.)

$$s =_{df} \{(d, g) \mid d = \infty, \dot{r} = 1, \dot{c}_1 = \dot{c}_2 = \dot{x}_1 = \dot{x}_2 = 0\},$$

with $g =_{df} r \times c_1 \times c_2 \times x_1 \times x_2$. We want to use the following statement:

“If the system is not in state *shutdown*, it must be in one of the other states.”

Using the set of *all* trajectories TRA we cannot characterise such a behaviour. Therefore we have to pick a subalgebra of TRA.

Lemma 6.1. *Let $A \subseteq \text{TRA}$ a set of trajectories. Then the structure*

$$\text{PRO}(A) =_{df} (\mathcal{P}(A^* \cup A^\omega), \cup, \emptyset, \cdot, I)$$

forms a Boolean left omega algebra.

To model liveness properties concerning the assembly line scheduler, we calculate in $\text{PRO}(l_1 \cup l_2 \cup i \cup s)$. The property that the system never reaches the state *shutdown* is now equivalent to the statement of never leaving the other states. The liveness property can be encoded as

$$(\mathbf{F} \cdot l_1)^\omega \leq (l_1 + l_2 + i)^\omega ,$$

where $\mathbf{F}$ denotes the set of all trajectories with finite duration. ($\mathbf{F}$ exists and can be defined in a general setting (e.g. [18,25]); here we only focus on applications and omit the theory.)

By coinduction and the hypothesis that $\mathbf{F} \leq (l_1 + l_2 + i)^*$ the claim follows immediately and can also be proved automatically. The hypothesis is by the additional assumption on s and can also be proved with Prover9 within 1 second. Details, like a proof by hand, can be found in Appendix A.

Therefore we have proved a liveness criterion for the assembly line scheduler.

7 Related Work

Although there is some related work concerning the verification of hybrid systems, we are not aware of any verification techniques based on first-order equational reasoning. But this is the key to using paramodulation-based first-order theorem provers.

Many verification techniques are based on hybrid automata [2]. But all these do not yield an algebraic approach; therefore no equation-based reasoning is possible. Furthermore, higher order theorem provers exist and are used to verify properties of hybrid systems. One of them is KeYmaera that extends the theorem prover KeY with Mathematica. It is a special purpose prover designed just for the verification of hybrid systems. Its advantage compared to our approach is that it also integrates arithmetic operators (see Section 8); but it needs a lot of interaction, since KeY is a higher-order prover. HyTech is a model-checker for hybrid systems. In [16] a preprocessor for HyTech is implemented which handles a limited version of LTL. A detailed comparison between that approach and our algebraic characterisation is still missing. A discussion on further related work is omitted for lack of space.

8 Conclusion and Outlook

In the paper we have shown that a trajectory-based algebra can be used to specify and verify safety and liveness properties. Algebraisation yields simple and short calculations. Moreover, these proofs can be automated with first-order theorem provers.

The presented work is only a first step of still on-going work. On the one hand the examples are still small. For that reason we want to do more case studies with larger systems. As a base we plan to use the examples of [6,26].

Although we have shown that the algebraic approach combined with first-order theorem proving is feasible, one still has to integrate arithmetics in our approach. So far we have derived preconditions by hand; namely the arithmetic constraints in the first example and the condition $F \leq (l_1 + l_2 + i)^*$ in the second. It would be interesting to see how this can be generalised and automated. At the moment we have two alternatives in mind: (1) There is some theory how to combine first-order theorem proving with arithmetics. In particular, for arithmetics based on integers there exists SPASS+T [27]. (2) In [16] HyTech is used to locally analyse hybrid systems. The outcome could be used to characterise and generate preconditions for our approach.

Acknowledgements. I am grateful to Georg Struth and Bernhard Möller for valuable remarks and discussions. Further I thank Martin Magnusson for discussions concerning the security service example.

References

1. Aboul-Hosn, K., Kozen, D.: KAT-ML: An interactive theorem prover for Kleene algebra with tests. *Journal of Applied Non-Classical Logics* 16(1–2), 9–33 (2006)
2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. *Theoretical Comp. Sc.* 138(1), 3–34 (1995)
3. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: *Hybrid Systems*, pp. 209–229. Springer, Heidelberg (1993)
4. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical Comp. Sc.* 126(2), 183–235 (1994)
5. Bergstra, J.A., Middleburg, C.A.: Process algebra for hybrid systems. *Theoretical Comp. Sc.* 335(2–3), 215–280 (2005)
6. Cho, K.-H., Johansson, K.H., Wolkenhauer, O.: A hybrid systems framework for cellular processes. *Biosystems* 80(3), 273–282 (2005)
7. Conway, J.H.: *Regular Algebra and Finite Machines*. Chapman & Hall, Sydney, Australia (1971)
8. Corbett, J.M.: Designing hybrid automated manufacturing systems: A european perspective. In: *Conference on Ergonomics of Hybrid Automated Systems I*, pp. 167–172. Elsevier, Amsterdam (1988)
9. Damm, W., Hungar, H., Olderog, E.-R.: On the verification of cooperating traffic agents. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.P. (eds.) *FMCO 2003. LNCS*, vol. 3188, pp. 77–110. Springer, Heidelberg (2004)
10. Davey, B.A., Priestley, H.A.: *Introduction to Lattices and Order*, 2nd edn. Cambridge University Press, Cambridge (2002)
11. Davoren, J.M., Nerode, A.: Logics for hybrid systems. *Proc. of the IEEE* 88(7), 985–1010 (2000)
12. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. *ACM Trans. Comp. Logic* 7(4), 798–833 (2006)

13. Faber, J., Meyer, R.: Model checking data-dependent real-time properties of the european train control system. In: FMCAD 2006, pp. 76–77. IEEE Press, Los Alamitos (2006)
14. Henzinger, T.A.: The theory of hybrid automata. In: Kemal, M. (ed.) IEEE Symposium on Logic in Computer Science (LICS 1996), pp. 278–292. IEEE Press, Los Alamitos (1996): Extended Version: Kemal, M.: Verification of Digital and Hybrid Systems. NATO ASI Series F: Computer and Systems Sciences, vol. 170, pp. 265–292. Springer, Heidelberg (2000)
15. Henzinger, T.A., Horowitz, B., Majumdar, R.: Rectangular hybrid games. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 320–335. Springer, Heidelberg (1999)
16. Henzinger, T.A., Majumdar, R.: Symbolic model checking for rectangular hybrid systems. In: Schwartzbach, M.I., Graf, S. (eds.) TACAS 2000. LNCS, vol. 1785, pp. 142–156. Springer, Heidelberg (2000)
17. Höfner, P., Möller, B.: Towards an algebra of hybrid systems. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) RelMiCS 2005. LNCS, vol. 3929, pp. 121–133. Springer, Heidelberg (2006)
18. Höfner, P., Möller, B.: An algebra of hybrid systems. Technical Report 2007-08, Institut für Informatik, Universität Augsburg (2007)
19. Höfner, P., Struth, G.: January 14 (2008), <http://www.dcs.shef.ac.uk/~georg/ka>
20. Höfner, P., Struth, G.: Automated reasoning in Kleene algebra. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 279–294. Springer, Heidelberg (2007)
21. Höfner, P., Struth, G.: Can refinement be automated? In: Boiten, E., Derrick, J., Smith, G. (eds.) Refine 2007. ENTCS, Elsevier, Amsterdam (to appear, 2007)
22. Kahl, W.: Calculational relation-algebraic proofs in Isabelle/Isar. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS 2003. LNCS, vol. 3051, pp. 179–190. Springer, Heidelberg (2004)
23. Kozen, D.: Kleene algebra with tests. Trans. Prog. Languages and Systems 19(3), 427–443 (1997)
24. McCune, W.: Prover9 and Mace4, <http://www.cs.unm.edu/~mccune/prover9>
25. Möller, B.: Kleene getting lazy. Sc. Comp. Prog. 65, 195–214 (2007)
26. Müller, O., Stauner, T.: Modelling and verification using linear hybrid automata – A case study. Math. and Comp. Modelling of Dynamical Systems 6(1), 71–89 (2000)
27. Prevosto, V., Waldmann, U.: SPASS+T. In: Sutcliffe, G., Schmidt, R., Schulz, S. (eds.) ESCoR: FLoC 2006, CEUR Workshop Proceedings, vol. 192, pp. 18–33 (2006)
28. Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press (2001)
29. Struth, G.: Calculating Church-Rosser proofs in Kleene algebra. In: de Swart, H. (ed.) RelMiCS 2001. LNCS, vol. 2561, pp. 276–290. Springer, Heidelberg (2002)
30. Sutcliffe, G., Yury, P.: SRASS-A semantic relevance axiom selection system. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 295–310. Springer, Heidelberg (2007)

A Omitted Details for the Assembly Line Scheduler

In the example of the assembly line scheduler all functions are real-valued, i.e., $r, c_1, c_2, x_1, x_2 : \mathbb{R} \rightarrow \mathbb{R}$; the set of durations is $\mathbb{R}$, too. The processes l_1 , l_2 , i and s are defined as follows:

$$\begin{aligned} l_1 &=_{df} \{(d, g) \mid \dot{r} = \dot{c}_2 = 1, \dot{c}_1 = \dot{x}_2 = 0, \dot{x}_1 = [1, 2]\} , \\ l_2 &=_{df} \{(d, g) \mid \dot{r} = \dot{c}_1 = 1, \dot{c}_2 = \dot{x}_1 = 0, \dot{x}_2 = [1, 2]\} , \\ i &=_{df} \{(d, g) \mid \dot{r} = \dot{c}_1 = \dot{c}_2 = 1, \dot{x}_1 = \dot{x}_2 = 0\} , \\ s &=_{df} \{(d, g) \mid d = \infty, \dot{r} = 1, \dot{c}_1 = \dot{c}_2 = \dot{x}_1 = \dot{x}_2 = 0\} , \end{aligned}$$

where g is defined as $g = r \times c_1 \times c_2 \times x_1 \times x_2$ and just collects all information of the behaviour. By coinduction, it is sufficient to show that $(F \cdot l_1)^\omega \leq (l_1 + l_2 + i)^* \cdot (F \cdot l_1)^\omega$. This follows from unfold, neutrality of 1, finiteness of 1 ($1 \leq F$), unfold again and the assumption:

$$(F \cdot l_1)^\omega = F \cdot l_1 \cdot (F \cdot l_1)^\omega \leq F \cdot F \cdot l_1 \cdot (F \cdot l_1)^\omega = F \cdot (F \cdot l_1)^\omega \leq (l_1 + l_2 + i)^* \cdot (F \cdot l_1)^\omega.$$

B Prover9 Source Code

Left omega algebras can be encoded in Prober9 as follows:

```
op(500, infix_left, "+"). %choice
op(490, infix_left, ";"). %composition
op(480, postfix, "*"). %finite iteration
op(450, postfix, "^"). %infinite iteration (omega)

formulas(sos).
% standard axioms of idempotent left semirings %%%%%%%%%%%
x+y = y+x. %commutative additive monoid
x+0 = x.
x+(y+z) = (x+y)+z. %multiplicative monoid
x;1 = x & 1;x = x.
x;(y;z) = (x;y);z.
0;x = 0. %annihilation laws
x+x = x. %idempotence
(x+y);z = x;z+y;z. %distributivity

% standard axioms for finite iteration (star) %%%%%%%%%%%
1+x;x* = x*.
(x;y+z)+y=y -> x*;z+y=y.

% standard axioms for infinite iteration (omega) %%%%%%%%%%%
x;x^ = x^.
y+(x;y+z)=x;y+z -> y+(x^+x*;z)=x^+x*;z.

end_of_list.

formulas(goals).
%lemma to be proved
end_of_list.
```

There exist also other implementations, e.g. an inequational encoding. They can be found at our website, too.

Non-termination in Idempotent Semirings

Peter Höfner and Georg Struth

Department of Computer Science
University of Sheffield, United Kingdom
`{p.hoefner,g.struth}@dcs.shef.ac.uk`

Abstract. We study and compare two notions of non-termination on idempotent semirings: infinite iteration and divergence. We determine them in various models and develop conditions for their coincidence. It turns out that divergence yields a simple and natural way of modelling infinite behaviour, whereas infinite iteration shows some anomalies.

1 Introduction

Idempotent semirings and Kleene algebras have recently been established as foundational structures in computer science. Initially conceived as algebras of regular expressions, they now find widespread applications ranging from program analysis and semantics to combinatorial optimisation and concurrency control.

Kleene algebras provide operations for modelling actions of programs or transition systems under non-deterministic choice, sequential composition and finite iteration. They have been extended by omega operations for infinite iteration [2,16], by domain and modal operators [4,12] and by operators for program divergence [3]. The resulting formalisms bear strong similarities with propositional dynamic logics, but have a much richer model class that comprises relations, paths, languages, traces, automata and formal power series.

Among the most fundamental analysis tasks for programs and reactive systems are termination and non-termination. In a companion paper [3], different algebraic notions of termination based on modal semirings have been introduced and compared. The most important ones are the omega operator for infinite iteration [2] and the divergence operator which models that part of a state space from which infinite behaviour may arise. Although, intuitively, absence of divergence and that of infinite iteration should be the same concept, it was found that they differ on some very natural models, including languages.

Here, we extend this investigation to the realm of non-termination. Our results further confirm the anomalies of omega. They also suggest that the divergence semirings proposed in [3] are powerful tools that capture terminating and non-terminating behaviour on various standard models of programs and reactive systems; they provide the right level of abstraction for analysing them in simple and concise ways. Our main contributions are as follows.

- We systematically compare infinite iteration and divergence in concrete models, namely finite examples, relations, traces, languages and paths. The concepts coincide in relation semirings, but differ on all other models considered.

- We also study abstract *taming conditions* for omega that imply coincidence with divergence. We find a rather heterogenous situation: Omega is tame on relation semirings. It is also tame on language semirings, but violates the taming condition. Therefore, the taming condition is only sufficient, but not necessary. In particular, omega is not tame on trace and path semirings.

The approach uses general results about fixed points for characterising and computing iterations in concrete models. Standard techniques from universal algebra relate the infinite models by Galois connections and homomorphisms.

All proofs at the level of Kleene algebras have been done by the automated theorem prover Prover9 [10]. They are documented at a website [7] and can easily be reproduced using the template in Appendix A. Proofs that use properties of particular models are given in Appendix B.

2 Idempotent Semirings and Omega Algebras

Our algebraic analysis of non-termination is based on idempotent semirings.

A *semiring* is a structure $(S, +, \cdot, 0, 1)$ such that $(S, +, 0)$ is a commutative monoid, $(S, \cdot, 1)$ is a monoid, multiplication distributes over addition and 0 is a left and right zero of multiplication. A semiring S is *idempotent* (an *i-semiring*) if $(S, +)$ is a semilattice with $x + y = \sup(x, y)$. (See the Prover9 input files in Appendix A for the axioms).

Idempotent semirings are useful for modelling actions, programs or state transitions under non-deterministic choice and sequential composition. We usually omit the multiplication symbol. The semilattice-order $\leq$ on S has 0 as its least element; addition and multiplication are isotone with respect to it.

Tests of a program or sets of states of a transition system can also be modelled in this setting. A *test* in an i-semiring S is an element of a Boolean subalgebra $\text{test}(S) \subseteq S$ (the *test algebra* of S) such that $\text{test}(S)$ is bounded by 0 and 1 and multiplication coincides with lattice meet. We will write $a, b, c, \dots$ for arbitrary semiring elements and $p, q, r, \dots$ for tests. We will freely use the standard laws of Boolean algebras on tests.

Iteration can be modelled on i-semirings by adding two operations.

A *Kleene algebra* [9] is an i-semiring S extended by an operation $*$: $S \rightarrow S$ that satisfies the *star unfold* and *star induction* axioms

$$1 + aa^* \leq a^*, \quad 1 + a^*a \leq a^*, \quad b + ac \leq c \Rightarrow a^*b \leq c, \quad b + ca \leq c \Rightarrow ba^* \leq c.$$

An *omega algebra* [2] is a Kleene algebra S extended by an operation $^\omega$: $S \rightarrow S$ that satisfies the *omega unfold* and the *omega co-induction* axiom

$$a^\omega \leq aa^\omega, \quad c \leq b + ac \Rightarrow c \leq a^\omega + a^*b.$$

a^*b and $a^\omega + a^*b$ are the least and the greatest fixed point of $\lambda x. b + ax$. The least fixed point of $\lambda x. 1 + ax$ is a^* and a^ω is the greatest fixed point of $\lambda x. ax$.

The star and the omega operator are intended to model finite and infinite iteration on i-semirings; Kleene algebras and omega algebras are intended as

algebras of regular and ω -regular events. A particular strength is that they allow first-order equational reasoning and therefore automated deduction [8]. Since i-semirings are an equational class, they are, by Birkhoff's HSP-theorem, closed under subalgebras, direct products and homomorphic images. Furthermore, since Kleene algebras and omega algebras are universal Horn classes, they are, by further standard results from universal algebra, closed under subalgebras and direct products, but not in general under homomorphic images. We will use these facts for constructing new algebras from given ones. Finite equational axiomatisations of algebras of regular events are ruled out since Kleene algebras are (sound and) complete for the equational theory of regular expressions, but there is no finite equational axiomatisation for this theory [9].

Consequently, all regular identities hold in Kleene algebras and we will freely use them. Examples are $0^* = 1 = 1^*$, $1 \leq a^*$, $aa^* \leq a^*$, $a^*a^* = a^*$, $a \leq a^*$, $a^*a = aa^*$ and $1 + aa^* = a^* = 1 + a^*a$. Furthermore the star is isotone.

It has also been shown that ω -regular identities such as $0^\omega = 0$, $a \leq 1^\omega$, $a^\omega = a^\omega 1^\omega$, $a^\omega = aa^\omega$, $a^\omega b \leq a^\omega$, $a^*a^\omega = a^\omega$ and $(a + b)^\omega = (a^*b)^\omega + (a^*b)^*a^\omega$ hold in omega algebras and that omega is isotone. Automated proofs of all these identities can be found at our website [7]. However, omega algebras are not complete for the equational theory of ω -regular expressions: Products of the form ab exist in ω -regular languages only if a represents a set of finite words whereas no such restriction is imposed on omega algebra terms. Moreover, every omega algebra has a greatest element $\top = 1^\omega$, and the following property holds [7].

$$(a + p)^\omega = a^\omega + a^*p\top. \quad (1)$$

3 Iterating Star and Omega

We will consider several important models in which a^* and a^ω do exist and in which a^* can be determined by fixed point iteration via the Knaster-Tarski theorem, whereas a^ω could only exist under additional assumptions that do not generally hold in our context. We will now set up the general framework.

One way to guarantee the existence of a^* and a^ω is to assume a *complete* i-semiring, i.e., an i-semiring with a complete semilattice reduct. Since every complete semilattice is also a complete lattice, a^* and a^ω exist and a^* can be approximated by $\sup(a^i : i \in \mathbb{N}) \leq a^*$ along the lines of Knaster-Tarski, where $\sup$ denotes the supremum operator. An iterative computation of a^*b presumes the additional infinite distributivity law

$$\sup(a^i : i \in \mathbb{N})b = \sup(a^ib : i \in \mathbb{N})$$

and similarly for ba^* . Such infinite laws always hold when the lattice reduct of the i-semiring is complete, Boolean, and meet coincides with multiplication. In particular, all *finite* i-semirings and all i-semirings defined on powersets with multiplication defined via pointwise extension are complete and the infinite distributivity laws hold. In all these cases, a^* can be iteratively determined as

$$a^* = \sup(a^i : i \in \mathbb{N})$$

and a^* is the reflexive transitive closure of a . Alternatively, the connection of a^* and iteration via suprema could be enforced by continuity [9].

It would be tempting to conjecture a dual iteration for a^ω . This would, however, presuppose distributivity of multiplication over arbitrary infima, which is not the case (cf. [13] for a counterexample). In general, we can only expect that

$$a^\omega \leq \inf(a^i \top : i \in \mathbb{N}).$$

An exception is the finite case, where every isotone function is also co-continuous. In this particular case, therefore $a^\omega = \inf(a^i \top : i \in \mathbb{N})$, i.e., a^ω can be iterated from the greatest element of a finite omega-algebra.

We will now illustrate the computation of star and omega in a simple finite relational example. This example will also allow us to motivate some concepts and questions that are treated in later sections.

Example 3.1. Consider the binary relation a in the first graph of Figure 1.

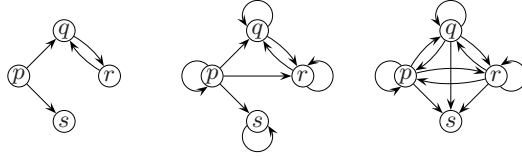


Fig. 1. The relations a , a^* and a^ω

Iterating $a^* = \sup(a^i : i \in \mathbb{N})$ yields the second graph of Figure 1. a^* represents the finite a -paths by collecting their input and output points: $(x, y) \in a^*$ iff there is a finite a -path from x to y .

Analogously one might expect that a^ω represents infinite a -paths in the sense that $(x, y) \in a^\omega$ iff x and y lie on an infinite a -path. However, iterating $a^\omega = \inf(a^i \top : i \in \mathbb{N})$ yields the right-most graph of Figure 1. It shows that $(q, p) \in a^\omega$ although there is no a -path from q to p , neither finite nor infinite.

So what does a^ω represent? Let ∇a model those nodes from which a diverges, i.e., from which an infinite a -path emanates. Then Example 3.1 shows that elements in ∇a are linked by a^ω to any other node; elements outside of ∇a are not in the domain of a^ω . Interpreting a^ω generally as *anything for states on which a diverges* would be consistent with the demonic semantics of total program correctness; its interpretation of *nothing for states on which a diverges* models partial correctness. This suggests to further investigate the properties

$$(\nabla a) \top = a^\omega \quad \text{and} \quad \nabla a = \text{dom}(a^\omega).$$

These two identities do not only hold in Example 3.1; they will be of central interest in this paper. To study them further, we will now introduce some important models of i-semirings and then formalise divergence in this setting.

4 Omega on Finite Idempotent Semirings

We have explicitly computed the stars and omegas for some small finite models using the model generator Mace4 [10]. We will further analyse these models in Section 9 and use them as counterexamples in Section 10.

Example 4.1. The two-element Boolean algebra is an i-semiring and an omega algebra with $0^* = 1^* = 1^\omega = 1$ and $0^\omega = 0$. It is the only two-element omega algebra and denoted by A_2 .

Example 4.2. There are three three-element i-semirings. Their elements are from $\{0, a, 1\}$. Only a is free in the defining tables. Stars and omegas are fixed by $0^* = 1^* = 1$, $0^\omega = 0$ and $1^\omega = \top$ (the greatest element) except for a .

- (a) In A_3^1 , addition is defined by $0 < 1 < a$, moreover, $aa = a^* = a^\omega = a$.
- (b) In A_3^2 , $0 < a < 1$, $aa = a^\omega = 0$ and $a^* = 1$.
- (c) In A_3^3 , $0 < a < 1$, $aa = a^\omega = a$ and $a^* = 1$.

5 Trace, Path and Language Semirings

We now present some of the most interesting models of i-semirings: traces, paths and languages. These are well-known; we formally introduce them only since we will study divergence and omega on these models in later sections.

As usual, a *word* over a set Σ is a mapping $[0..n] \rightarrow \Sigma$. The empty word is denoted by ε and *concatenation* of words σ_0 and σ_1 by $\sigma_0.\sigma_1$. We write $\text{first}(\sigma)$ for the first element of a word σ and $\text{last}(\sigma)$ for its last element. We write $|\sigma|$ for the length of σ . The set of all words over Σ is denoted by Σ^* .

A (finite) *trace* over the sets P and A is either ε or a word σ such that $\text{first}(\sigma)$, $\text{last}(\sigma) \in P$ and in which elements from P and A alternate. $\tau_0, \tau_1, \dots$ will denote traces. For $s \in P$ the product of traces τ_0 and τ_1 is the trace

$$\tau_0 \cdot \tau_1 = \begin{cases} \sigma_0.s.\sigma_1 & \text{if } \tau_0 = \sigma_0.s \text{ and } \tau_1 = s.\sigma_1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Intuitively, $\tau_0 \cdot \tau_1$ glues two traces together when the last state of τ_0 and the first state of τ_1 are equal. The set of all traces over P and A is denoted by $(P, A)^*$, where P is the set of *states* and A the set of *actions*.

Lemma 5.1. *The power-set algebra $2^{(P,A)^*}$ with addition defined by set union, multiplication by $S \cdot T = \{\tau_0 \cdot \tau_1 : \tau_0 \in S, \tau_1 \in T \text{ and } \tau_0 \cdot \tau_1 \text{ defined}\}$, and with $\emptyset$ and P as neutral elements is an i-semiring.*

We call this i-semiring the *full trace semiring* over P and A . By definition, $S \cdot T = \emptyset$ if all products between traces in S and traces in T are undefined.

Every subalgebra of the full trace semiring is, by the HSP-theorem, again an i-semiring (constants such as 0, 1 and $\top$ are fixed by subalgebra constructions). We will henceforth consider only complete subalgebras of full trace semirings

and call them *trace semirings*. Every non-complete subalgebra of the full trace semiring can of course uniquely be closed to a complete subalgebra.

As we will see, forgetting parts of the structure is quite useful. First we want to forget all actions of traces. Consider the *projection* $\phi_P : (P, A)^* \rightarrow P^*$ which is defined, for all $s \in P$ and $\alpha \in A$, by

$$\phi_P(\varepsilon) = \varepsilon, \quad \phi_P(s.\sigma) = s.\phi_P(\sigma), \quad \phi_P(\alpha.\sigma) = \phi_P(\sigma).$$

ϕ_P is a mapping between traces and words over P which we call *paths*. Moreover it can be seen as the homomorphic extension of the function $\phi(\varepsilon) = \phi(\alpha) = \varepsilon$ and $\phi(s) = s$ with respect to concatenation. A product on paths can be defined as for traces. Again, $\pi_0 \cdot \pi_1$ glues two paths π_0 and π_1 together when the last state of π_0 and the first state of π_1 are equal.

The mapping ϕ_P can be extended to a set-valued mapping $\phi_P : 2^{(P,A)^*} \rightarrow 2^{P^*}$ by taking the image, i.e., $\phi_P(T) = \{\phi_P(\tau) : \tau \in T\}$. Now, ϕ_P sends sets of traces to sets of paths.

The information about actions can be introduced to paths by *fibration*, which can be defined in terms of the relational inverse $\phi_P^{-1} : P^* \rightarrow 2^{(P,A)^*}$ of ϕ_P . Intuitively, it fills the spaces between states in a path with all possible actions and therefore maps a single path to a set of traces. The mapping ϕ_P^{-1} can as well be lifted to the set-valued mapping $\phi_P^\sharp(Q) = \sup(\phi_P^{-1}(\pi) : \pi \in Q)$, where $Q \in 2^{P^*}$ is a set of paths.

Lemma 5.2. ϕ_P and $\phi_P^\sharp$ are adjoints of a Galois connection, i.e., for $a \in 2^{(P,A)^*}$ and $b \in 2^{P^*}$ we have

$$\phi_P(a) \leq b \Leftrightarrow a \leq \phi_P^\sharp(b).$$

The proof of this fact is standard. Galois connections are interesting because they give theorems for free. In particular, ϕ_P commutes with all existing suprema and $\phi_P^\sharp$ commutes with all existing infima. Also, ϕ_P is isotone and $\phi_P^\sharp$ is antitone. Both mappings are related by the cancellation laws $\phi_P \circ \phi_P^\sharp \leq id_{2^{P^*}}$ and $id_{2^{(P,A)^*}} \leq \phi_P^\sharp \circ \phi_P$. Finally, the mappings are pseudo-inverses, that is, $\phi_P \circ \phi_P^\sharp \circ \phi_P = \phi_P$ and $\phi_P^\sharp \circ \phi_P \circ \phi_P^\sharp = \phi_P^\sharp$.

Lemma 5.3. The mappings ϕ_P are homomorphisms.

By the HSP-theorem the set-valued homomorphism induces path semirings from trace semirings.

Lemma 5.4. The power-set algebra 2^{P^*} is an i-semiring.

We call this i-semiring the *full path semiring* over P . It is the homomorphic image of a full trace semiring. Again, by the HSP-theorem, all subalgebras of full path semirings are i-semirings; complete subalgebras are called *path semirings*.

Lemma 5.5. Every identity that holds in all trace semirings holds in all path semirings.

Moreover, the class of trace semirings contains isomorphic copies of all path semirings. This can be seen as follows.

Consider the congruence $\sim_P$ on a trace semiring over P and A that is induced by the homomorphism ϕ_P . The associated equivalence class $[T]_P$ contains all those sets of traces that differ in actions, but not in paths. From each equivalence class we can choose as canonical representative a set of traces all of which are built from one single action. Each of these representatives is of course equivalent to a set of paths and therefore an element of a path semiring. Conversely, every element of a path semiring can be expanded to an element of some trace semiring by filling in the same action between all states.

The following lemma can be proved using techniques from universal algebra.

Lemma 5.6. *Let S be the full trace semiring over P and A . The quotient algebra $S/\sim_P$ is isomorphic to each full trace semiring over P and $\{a\}$ with $a \in A$ and to the full path semiring over P :*

$$S/\sim_P \cong 2^{(P, \{a\})^*} \cong 2^{P^*}.$$

In particular, the mappings ϕ_P and $\phi_P^\sharp$ are isomorphisms between the full trace semiring $2^{(P, \{a\})^*}$ and the full path semiring 2^{P^*} . In that case, $\phi_P^{-1} = \phi_P^\sharp$.

Lemma 5.6 is not only limited to full trace and path semirings. It immediately extends to trace and path semirings, since the operations of forming subalgebras and of taking homomorphic images always commute. In particular, each path semiring is isomorphic to some trace semiring with a single action. This isomorphic embedding of path semirings into the class of trace semirings implies the following proposition.

Proposition 5.7. *Every first-order property that holds in all trace semirings holds in all path semirings.*

In particular, Horn clauses that hold in all trace semirings are also valid in the setting of paths.

A similar mapping and Galois connection for languages can be defined by forgetting states, but it does not extend to a homomorphism: forgetting states before or after products yields different results. Nevertheless, the class of trace semirings contains again elements over one single state. These are isomorphic to (complete) *language semirings*, which are algebras of formal languages. Conversely, every language semiring can be induced by this isomorphism.

Proposition 5.8. *Every first-order property that holds in all trace semirings holds in all language semirings.*

6 Relation Semirings

Now we forget entire paths between the first and the last state of a trace. We therefore consider the mapping $\phi_R : (P, A)^* \rightarrow P \times P$ defined by

$$\phi_R(\tau) = \begin{cases} (\text{first}(\tau), \text{last}(\tau)) & \text{if } \tau \neq \varepsilon, \\ \text{undefined} & \text{if } \tau = \varepsilon. \end{cases}$$

It sends trace products to (standard) relational products on pairs. As before, ϕ_R can be extended to a set-valued mapping $\phi_R : 2^{(P,A)^*} \rightarrow 2^{P \times P}$ by taking the image, i.e., $\phi_R(T) = \{\phi_R(\tau) : \tau \in T\}$. Now, ϕ_R sends sets of traces to *relations*. Information about the traces between starting and ending states can be introduced to pairs of states by the fibration $\phi_R^{-1} : P \times P \rightarrow 2^{(P,A)^*}$ of ϕ_R . Intuitively, it replaces a pair of states by all possible traces between them. It can again be lifted to the set-valued mapping $\phi_R^\sharp(R) = \sup(\phi_R^{-1}(r) : r \in R)$, for any relation $R \in 2^{P \times P}$.

Lemma 6.1. *ϕ_R and $\phi_R^\sharp$ are adjoints of a Galois connection.*

The standard properties hold again.

Lemma 6.2. *The mappings ϕ_R are homomorphisms.*

By the HSP-theorem, the set-valued homomorphism induces relation semirings from trace semirings.

Lemma 6.3. *The power-set algebra $2^{P \times P}$ is an i-semiring.*

We call this i-semiring the *full relation semiring* over P . It is the homomorphic image of a full trace semiring. Again, all subalgebras of full relation semirings are i-semirings; complete subalgebras are called *relation semirings*.

Proposition 6.4. *Every identity that holds in all trace semirings holds in all relation semirings.*

Similar to $\sim_P$ we can define $\sim_R$ induced by ϕ_R . But in that case, multiplication is not well-defined in general and the quotient structures induced are not semirings.

Lemma 6.5. *There is no trace semiring over P and A that is isomorphic to the full relation semiring over a finite set Q with $|Q| > 1$.*

A homomorphism that sends path semirings to relation semirings can be built in the same way as ϕ_R and $\phi_R^\sharp$, but using paths instead of traces as an input. The homomorphism $\chi : 2^{A^*} \rightarrow 2^{A^* \times A^*}$ that sends language semirings to relation semirings uses a standard construction (cf. [14]). It is defined, for all $L \subseteq A^*$ by $\tilde{\chi}(L) = \{(v, v.w) : v \in A^* \text{ and } w \in L\}$.

Lemma 6.6. *Every identity that holds in all path or language semirings holds in all relation semirings.*

It is important to distinguish between relation semirings and relational structures under addition and multiplication in general.

We will often need to consider trace semirings and relation semirings separately, whereas language and path semirings are subsumed.

7 Omega on Trace, Language and Path Semirings

Let us consider star and omega in (infinite) trace, path and language semirings. We will relate the results obtained with divergence in Section 9. We will also study omega and divergence on relation semirings in that section.

We first consider trace semirings. By definition, they are complete and satisfy all necessary infinite distributivity laws. Stars can therefore be determined by iteration, omegas cannot.

A sets of traces S over P and A can always be partitioned in its *test part* $S_t = S \cap P$ and its *test-free* or *action part* $S_a = S - P$, i.e., $S = S_t + S_a$. This allows us to calculate S_a^ω separately and then to combine them by Equation (II) to $S^\omega = S_a^\omega + S_a^* S_t^\omega$.

Since S_a is test-free, every trace $\tau \in S_a^\omega$ satisfies $|\tau| > 1$. Therefore, by induction, $|\tau| > n$ for all $\tau \in S_a^n$ and consequently $S_a^\omega = \inf(S_a^i : i \in \mathbb{N}) = \emptyset$.

As a conclusion, in trace models omega can be explicitly defined by the star. This might be surprising: Omega, which seemingly models infinite iteration, reduces to finite iteration after which a miracle (*anything*) happens. By the results of the previous sections, the argument also applies to language and path semirings. In the case of languages, the argument is known as *Arden's rule* [II]. In particular, the test algebras of language algebras are always $\{\emptyset, \{\varepsilon\}\}$. Therefore $L^\omega = \emptyset$ iff $\varepsilon \notin L$ for every language $L \in 2^{A^*}$.

Theorem 7.1. *Assume an arbitrary element a of $2^{(P,A)^*}$, 2^{A^*} and 2^{P^*} , respectively. Let $a_t = a \cap 1$ denote the test and $a_a = a - a_t$ the action part of a .*

- (a) *In trace semirings, $a^\omega = (a_a)^* a_t^\omega$ for any $a \in 2^{(P,A)^*}$.*
- (b) *In language semirings, $a^\omega = A^*$ if $\varepsilon \in a$ and $\emptyset$ otherwise for any $a \in 2^{A^*}$.*
- (c) *In path semirings, $a^\omega = a^* a_t^\omega$ for any $a \in 2^{P^*}$.*

In relation semirings the situation is different: there is no notion of length that would increase through iteration. We will therefore determine omegas in relation semirings relative to a notion of divergence (cf. Section 9).

8 Divergence Semirings

An operation of divergence can be axiomatised algebraically on i-semirings with additional modal operators. The resulting divergence semirings are similar to Goldblatt's *foundational algebras* [6].

An i-semiring S is called *modal* [12] if it can be endowed with a total operation $\langle a \rangle : \text{test}(S) \rightarrow \text{test}(S)$, for each $a \in S$, that satisfies the axioms

$$\langle a \rangle p \leq q \Leftrightarrow ap \leq qa \quad \text{and} \quad \langle ab \rangle p = \langle a \rangle \langle b \rangle p.$$

Intuitively, $\langle a \rangle p$ characterises the set of states with at least one a -successor in p . A *domain* operation $\text{dom} : S \rightarrow \text{test}(S)$ is obtained from the diamond operator as $\text{dom}(a) = \langle a \rangle 1$. Alternatively, domain can be axiomatised on i-semirings, even equationally, from which diamonds are defined as $\langle a \rangle p = \text{dom}(ap)$ [3]. The axiomatisation of modal semirings extends to modal Kleene algebras and modal omega algebras without any further modal axioms.

We will use the following properties of diamonds and domain [7]: $\langle p \rangle q = pq$, $\text{dom}(a) = 0 \Leftrightarrow a = 0$, $\text{dom}(\top) = 1$, $\text{dom}(p) = p$. Also, domain is isotone and diamonds are isotone in both arguments.

A modal semiring S is a *divergence semiring* [3] if it has an operation $\nabla : S \rightarrow \text{test}(S)$ that satisfies the ∇ -unfold and ∇ -co-induction axioms

$$\nabla a \leq \langle a \rangle \nabla a \quad \text{and} \quad p \leq \langle a \rangle p \Rightarrow p \leq \nabla a.$$

We call ∇a the *divergence* of a . This axiomatisation can be motivated on trace semirings as follows: The test $p - \langle a \rangle p$ characterises the set of a -maximal elements in p , that is, the set of elements in p from which no further a -action is possible. ∇a therefore has no a -maximal elements by the ∇ -unfold axiom and by the ∇ -co-induction axiom it is the greatest set with that property. It is easy to see that $\nabla a = 0$ iff a is Noetherian in the usual set-theoretic sense. Divergence therefore comprises the standard notion of program termination. All those states that admit only finite traces are characterised by the complement of ∇a .

The ∇ -co-induction axiom is equivalent to $p \leq q + \langle a \rangle p \Rightarrow p \leq \nabla a + \langle a^* \rangle q$, which has the same structure as the omega co-induction axiom. In particular, ∇a is the greatest fixed point of the function $\lambda x. \langle a \rangle x$, which corresponds to a^ω and $\nabla a + \langle a^* \rangle q$ is the greatest fixed point of the function $\lambda x. q + \langle a \rangle x$, which corresponds to $a^\omega + a^*b$. Moreover, the least fixed point of $\lambda x. q + \langle a \rangle x$ is $\langle a^* \rangle q$, which corresponds to a^*b . These fixed points are now defined on test algebras, which are Boolean algebras. Iterative solutions exist again when the test algebra is finite and all diamonds are defined. In general

$$\nabla a \leq \inf(\langle a^i \rangle 1 : i \in \mathbb{N}) = \inf(\text{dom}(a^i) : i \in \mathbb{N}).$$

However, the algebra A_3^2 shows that even finite i-semirings, which always have a complete test algebra, need not be modal semirings (cf. Example 9.2 below).

We will need the properties $\langle a \rangle \nabla a \leq \nabla a$, $\nabla p = p$ and $\nabla a \leq \text{dom}(a)$ of divergence and isotonicity of ∇ [7].

9 Divergence Across Models

We will now relate omega and divergence in all models presented so far. Concretely, we will investigate the identities $(\nabla a)\top = a^\omega$ and $\nabla a = \text{dom}(a^\omega)$ that arose from our motivating example in Section 3. We will say that omega is *tame* if every a satisfies the first identity; it will be called *benign* if every a satisfies the second one. We will also be interested in the *taming condition* $\text{dom}(a)\top = a\top$. All abstract results of this and the next section have been again automatically verified by Prover9 or Mace4.

First, we consider these properties on relation semirings which we could not treat as special cases of trace semirings in Section 7. It is well known from relation algebra that all relation semirings satisfy the taming condition. We will see in the following section through abstract calculations that omega and divergence are related in relation semirings as expected and, as a special case, $a^\omega = 0$ iff a is Noetherian in relation semirings.

We now revisit the finite i-semirings of Examples 4.1 and 4.2.

Example 9.1. In A_2 , $\text{dom}(0) = 0$ and $\text{dom}(1) = 1$. By this, $\nabla 0 = 0$ and $\nabla 1 = 1$.

Example 9.2. In A_3^1 and A_3^3 , the test algebra is always $\{0, 1\}$; $\text{dom}(0) = 0$ and $\text{dom}(1) = 1$. Moreover, $\nabla 0 = 0$ and $\nabla 1 = 1$. Setting $\text{dom}(a) = 1 = \nabla a$ turns both into divergence semirings. In contrast, domain cannot be defined on A_3^2 .

Consequently, omega is not tame in A_3^2 , since $\nabla a \top$ is undefined here, and in A_3^3 . However, it is tame in A_3^1 and A_2 . In all four finite i-semirings, omega is benign.

Let us now consider trace, language and path semirings. Domain, diamond and divergence can indeed be defined on all these models. On a trace semiring,

$$\text{dom}(S) = \{s : s \in P \text{ and } \exists \tau \in (P, A)^* : s \cdot \tau \in S\}.$$

So, as expected, $\nabla S = \inf(\text{dom}(S^i) : i \in \mathbb{N})$; it characterises all states where infinite paths may start. However, since the omega operator is related to finite behaviour in all these models (cf. Theorem 7.1), the expected relationships to divergence fail.

Lemma 9.3. *The taming condition does not hold on some trace and path semirings. Omega is neither tame nor benign.*

The situation for language semirings, where states are forgotten, is different.

Lemma 9.4

- (a) *The taming condition does not hold in some language semirings.*
- (b) *Omega is tame in all language semirings.*
- (c) *$(\nabla a) \top = a^\omega \not\Rightarrow \text{dom}(a) \top = a \top$ in some language semirings.*

In the next section we will provide an abstract argument that shows that omega is benign on language semirings (without satisfying the taming condition).

As a conclusion, omega behaves as expected in relation semirings, but not in trace, language and path semirings. This may be surprising: While relations are standard for finite input/output behaviour, traces, languages and paths are standard for infinite behaviour, including reactive and hybrid systems. As we showed before, in these models omega can be expressed by the finite iteration operator and therefore it does not model proper infinite iteration. In contrast to that the divergence operator models infinite behaviour in a natural way.

10 Taming the Omega

Our previous results certainly deserve a model-independent analysis. We henceforth briefly call *omega divergence semirings* a divergence semiring that is also an omega algebra. We will now consider tameness of omega for this class. It is easy to show that the simple identities

$$a \top \leq \text{dom}(a) \top, \quad a^\omega \leq (\nabla a) \top, \quad \text{dom}(a^\omega) \leq \nabla a,$$

hold in all omega divergence semirings [7]. Therefore we only need to consider the relationships between their converses.

Theorem 10.1. *In the class of omega divergence semirings, the following implications hold, but not their converses.*

$$\begin{aligned}\forall a. (\text{dom}(a)\top \leq a\top) &\Rightarrow \forall a. (\nabla a)\top \leq a^\omega, \\ (\nabla a)\top \leq a^\omega &\Rightarrow \nabla a \leq \text{dom}(a^\omega).\end{aligned}$$

Theorem 10.1 shows that the taming condition implies that omega is tame, which again implies that omega is benign. The fact that omega is benign whenever it satisfies the taming condition has already been proved in [3]. In particular, all relational semirings are tame and benign, since they satisfy the taming condition.

Theorem 10.1 concludes our investigation of divergence and omega. It turns out that these two notions of non-termination are unrelated in general. Properties that seem intuitive for relations can be refuted on three-element or natural infinite models. The taming condition that seems to play a crucial role could only be verified on (finite and infinite) relation semirings.

11 Conclusion

We compared two algebraic notions of non-termination: the omega operator and divergence. It turned out that divergence correctly models infinite behaviour on all models considered, whereas omega shows surprising anomalies. In particular, omega is not benign (whence not tame) on traces and paths, which are among the standard models for systems with infinite behaviour such as reactive and hybrid systems. A particular advantage of our algebraic approach is that this analysis could be carried out in a rather abstract, uniform and simple way.

The main conclusion of this paper, therefore, is that idempotent semirings are a very useful tool for reasoning about termination and infinite behaviour across different models. The notion of divergence is a simple but powerful concept for representing that part of a state space at which infinite behaviour may start. The impact of this concept on the analysis of discrete dynamical systems, in particular by automated reasoning, remains to be explored. The omega operator, however, is appropriate only under some rather strong restrictions which eliminate many models of interest. Our results clarify that omega algebras are generally inappropriate for infinite behaviour: It seems unreasonable to sequentially compose an infinite element a with another element b to ab . Two alternatives to omega algebras allow adding infinite elements: The weak variants of omega algebras introduced by von Wright [16] and elaborated by Möller [11], and in particular the divergence modules introduced in [15], based on work of Ésik and Kuich [5], in which finite and infinite elements have different sorts and divergence is a mapping from finite to infinite elements. All these variants are developed within first-order equational logic and therefore support the analysis of infinite and terminating behaviours of programs and transition systems by automated deduction [15]. The results of this paper link this abstract analysis with properties of particular models which may arise as part of it.

Acknowledgement. We are grateful to Bernhard Möller for proof-reading.

References

1. Arden, D.: Delayed logic and finite state machines. In: Theory of Computing Machine Design, pp. 1–35. University of Michigan Press (1960)
2. Cohen, E.: Separation and reduction. In: Backhouse, R., Oliveira, J.N. (eds.) MPC 2000. LNCS, vol. 1837, pp. 45–59. Springer, Heidelberg (2000)
3. Desharnais, J., Möller, B., Struth, G.: Termination in modal Kleene algebra. In: Lévy, J.-J., Mayr, E.W., Mitchell, J.C. (eds.) IFIP TCS 2004, pp. 647–660. Kluwer, Dordrecht (2004); Revised version: Algebraic Notions of Termination. Technical Report 2006-23, Institut für Informatik, Universität Augsburg (2006)
4. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM Trans. Computational Logic 7(4), 798–833 (2006)
5. Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω -context-free languages. In: Karhumäki, J., Maurer, H., Păun, G., Rozenberg, G. (eds.) Theory Is Forever. LNCS, vol. 3113, pp. 68–80. Springer, Heidelberg (2004)
6. Goldblatt, R.: An algebraic study of well-foundedness. Studia Logica 44(4), 423–437 (1985)
7. Höfner, P., Struth, G.: January 14 (2008), <http://www.dcs.shef.ac.uk/~georg/ka>
8. Höfner, P., Struth, G.: Automated reasoning in Kleene algebra. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 279–294. Springer, Heidelberg (2007)
9. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation 110(2), 366–390 (1994)
10. McCune, W.: Prover9 and Mace4, January 14 (2008), <http://www.cs.unm.edu/~mccune/prover9>
11. Möller, B.: Kleene getting lazy. Sci. Comput. Programming 65, 195–214 (2007)
12. Möller, B., Struth, G.: Algebras of modal operators and partial correctness. Theoretical Computer Science 351(2), 221–239 (2006)
13. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
14. Pratt, V.: Dynamic algebras: Examples, constructions, applications. Studia Logica 50, 571–605 (1991)
15. Struth, G.: Reasoning automatically about termination and refinement. In: Ranise, S. (ed.) International Workshop on First-Order Theorem Proving, Technical Report ULCS-07-018, Department of Computer Science, University of Liverpool, pp. 36–51 (2007)
16. von Wright, J.: Towards a refinement algebra. Science of Computer Programming 51(1-2), 23–45 (2004)

Appendices

A A Proof Template for Prover9

```

op(500, infix, "+").    %addition
op(490, infix, ";").    %multiplication
op(480, postfix, "*").  %star
op(470, postfix, "^").  %omega

formulas(sos).

% Kleene algebra axioms
x+y = y+x & x+0 = x & x+(y+z) = (x+y)+z.
x;(y;z) = (x;y);z & x;1 = x & 1;x = x.
0;x = 0 & x;0 = 0.
x;(y+z) = x;y+x;z & (x+y);z = x;z+y;z.
x+x = x.
x <= y <-> x+y = y.
1+x;x* = x* & 1+x*;x = x*.
z+x;y <= y -> x*;z <= y & z+y;x <= y -> z;x* <= y.

% Boolean domain axioms (a la Desharnais & Struth)
a(x);x = 0 & a(x;y) = a(x;a(a(y))) & a(a(x))+a(x) = 1.
d(x) = a(a(x)).    %domain defined from antidomain

% divergence
d(x;div(x)) = div(x).
d(y) <= d(x;d(y))+d(z) -> d(y) <= div(x)+d(x*;z).

% omega axioms
x;x^ = x^ & z <= x;z+y -> z <= x^+x*;y.

% additional laws
T = 1^.
x <= y -> d(x) <= d(y).
end_of_list.

formulas(goals). % for Thm 10.1; to be commented in one by one
%all x(d(x);T <= x;T) -> all x(div(x);T <= x^).
%div(x);T <= x^ -> div(x) <= d(x^).
%all x(d(x);T <= x;T) <- all x(div(x);T = x^).
%div(x);T <= x^ -> div(x) = d(x^).
end_of_list.

```

B Proofs

Lemma 6.5. *There is no trace semiring over P and A that is isomorphic to the full relation semiring over a finite set Q with $|Q| > 1$.*

Proof. If there is at least one action in the trace semiring, then the trace semiring is infinite whereas the size of the relation semiring is $2^{|Q|^2}$. Otherwise, all traces will be single states and multiplication will therefore commute on the trace semiring, but not on the relation semiring. Therefore there cannot exist an isomorphism. $\square$

Lemma 9.3. *The taming condition does not hold on some trace and path semirings. Omega is neither tame nor benign.*

Proof. Consider the case of trace semirings. Let $P = \{s\}$ and $A = \{\alpha\}$ and let S be the set consisting of the single trace $s\alpha s$. Then $\text{dom}(S) = \{s\} = \nabla S$ and $\text{dom}(S)\top = \{s\}\top = \nabla(S)\top$ is the set of all non-empty traces over p and α . Moreover, $S\top = \{s.\alpha.\tau : \tau \in (P, A)^*\}$. Finally, Theorem 7.1(a) implies that $S^\omega = S_a^* S_t \top = \emptyset$ since $S_t = \emptyset$ in the example. This refutes all identities for trace semirings. The argument translates to path semirings by forgetting actions. $\square$

Lemma 9.4

- (a) *The taming condition does not hold in some language semirings.*
- (b) *Omega is tame in all language semirings.*
- (c) *Tameness does not imply the taming condition in some language semirings.*

Proof. In language semirings the test algebra is $\{\emptyset, \{\varepsilon\}\}$. So $\text{dom}(L) = \{\varepsilon\}$ iff $L \neq 0$ for every $L \in 2^{A^*}$.

- (a) Consider the language semiring over the single letter a and the language $L = \{a\}$. Then $\text{dom}(L) = \{\varepsilon\}$ and therefore $\text{dom}(L)\top = \top \neq L\top$, since $\varepsilon \in \top$, but $\varepsilon \notin L\top$.
- (b) $\nabla L = \inf(\text{dom}(L^i) : i \in \mathbb{N}) = \{\varepsilon\}$ iff $L \neq \emptyset$. Therefore $(\nabla L)\top = \top$ iff $L \neq \emptyset$ and $(\nabla L)\top = \emptyset$ iff $L = \emptyset$. It has already been shown in Lemma 7.1(b) that L^ω satisfies the same conditions.
- (c) Immediate from (a) and (b). $\square$

Theorem 10.1. *In the class of omega divergence semirings, the following implications hold, but not their converses.*

$$\begin{aligned} \forall a. (\text{dom}(a)\top \leq a\top) &\Rightarrow \forall a. (\nabla a)\top \leq a^\omega, \\ (\nabla a)\top \leq a^\omega &\Rightarrow \nabla a \leq \text{dom}(a^\omega). \end{aligned}$$

Proof. Both implications can be proved in a few seconds by Prover9 on any personal computer with the input file from Appendix A.

The converse of the first implication fails in the class of language semirings by Lemma 9.4(c).

The converse of the second implication fails in A_3^3 since $\nabla a = 1 = \text{dom}(a) = \text{dom}(a^\omega)$ holds in this model, but $(\nabla a)\top = 1 > a = a^\omega$ by Example 4.2 and 9.2. $\square$

Formal Concepts in Dedekind Categories

Toshikazu Ishida, Kazumasa Honda, and Yasuo Kawahara

Department of Informatics, Kyushu University, Fukuoka, 819-0395, Japan
t-ishida@i.kyushu-u.ac.jp

Abstract. Formal concept analysis is a mathematical field applied to data mining. Usually, a formal concept is defined as a pair of sets, called extents and intents, for a given formal context in binary relation. In this paper we set a relational formulation for formal concepts, and prove some basic properties of concept lattices by using relation calculus in Dedekind categories.

1 Introduction

In data mining, we aim to discover hidden information, such as patterns and correlations, from massive data. In fact, the method is widely used in economic and scientific activities. Formal concept analysis [1] is a mathematical field proposed by R.Wille in 1970's. Based on lattice theory, it enables us to search logic and knowledge structures, such as patterns and correlations, by means of concept lattices obtained from database. Therefore, formal concept analysis is applicable to data mining.

The database consists of binary relation between objects and attributes. A formal concept is defined as a pair of sets objects and attributes which satisfy given conditions. The sets are called extents and intents.

In this paper we set a relational formulation [2] for formal concepts, and prove some basic properties of concept lattices by using relation calculus [3,4,5,6] in Dedekind categories. Our method, set away from the concept of elements in categories, could be applied to functional programming for formal concept analysis. Besides, by use of residual composition, the method could be extended to a fuzzy relation and a relation of multivalued logic.

This paper is composed as follows. In Section 2 and 3, we review the definitions and basic properties of formal concepts in set theory and Dedekind categories. In Section 4, we define membership relations and inclusion orders, and state their fundamental properties [7]. In Section 5, we define formal concepts in Dedekind categories. The basic theorems mentioned in Section 2 are proved by use of relation calculus. Finally, we show that a given formal context and its reduced context have the isomorphic concept lattice.

2 Formal Concept

In this section, we review the basic definitions and properties of formal concepts [1].

A formal context is a binary relation I between objects and attributes. A formal concept (A, B) of I satisfy that, “ B is the set of attributes that every

object of A connected by the relation I . And A is the set of objects that every attribute of B connected by the relation I .” We call A the *extent* and B the *intent* of the formal concept (A, B) . Additionally, a concept lattice is made from the set of all formal concepts of a formal context and order of concepts.

Definition 1. A *formal context* (X, Y, I) consists of two sets X and Y and a relation I between X and Y . The elements of X and Y are called the *objects* and the *attributes* of the formal context. $\square$

Definition 2. For a subset A of X , we define

$$A^\uparrow = \{y \in Y \mid (x, y) \in I \text{ for all } x \in A\}.$$

Correspondingly, for a subset B of Y , we define

$$B^\downarrow = \{x \in X \mid (x, y) \in I \text{ for all } y \in B\}. \quad \square$$

Definition 3. A *formal concept* of the formal context (X, Y, I) is a pair (A, B) of a subset A of X and a subset B of Y such that $A^\uparrow = B$ and $B^\downarrow = A$. We call A the *extent* and B the *intent* of the formal concept (A, B) . $\square$

Theorem 1. If (X, Y, I) is a formal context, $A, A_1, A_2 \subseteq X$ are sets of objects and $B, B_1, B_2 \subseteq Y$ are sets of attributes, then

$$\begin{array}{ll} \text{(a)} & A_1 \subseteq A_2 \implies A_1^\uparrow \subseteq A_2^\uparrow, & (B_1 \subseteq B_2 \implies B_1^\downarrow \subseteq B_2^\downarrow) \\ \text{(b)} & A \subseteq A^{\uparrow\downarrow}, & (B \subseteq B^{\downarrow\uparrow}) \\ \text{(c)} & A^\uparrow = A^{\uparrow\downarrow\uparrow}, & (B^\downarrow = B^{\downarrow\uparrow\downarrow}) \\ \text{(d)} & A \subseteq B^\downarrow \iff B \subseteq A^\uparrow. & \end{array} \quad \square$$

The set of all formal concepts $C(I)$ of the formal context (X, Y, I) is defined as follows:

$$C(I) = \{(A, B) \mid A^\uparrow = B \text{ and } B^\downarrow = A, \text{ for all } A \subseteq X, B \subseteq Y\}.$$

The set $C(I)$ is naturally ordered by the inclusion $\subseteq$ of objects.

Definition 4. If (A_1, B_1) and (A_2, B_2) are formal concepts, (A_1, B_1) is called *subconcept* of (A_2, B_2) , provided that $A_1 \subseteq A_2$. In this case, (A_2, B_2) is a *superconcept* of (A_1, B_1) , and we write $(A_1, B_1) \leq (A_2, B_2)$. The relation $\leq$ is called the *order* of formal concepts. The set of all formal concepts of (X, Y, I) ordered in this way is denoted by $C(I)$ and the ordered set $(C(I), \leq)$ is called the *concept lattice* of the formal context (X, Y, I) . $\square$

The following theorem shows the reason why $C(I)$ is a lattice.

Theorem 2. Let T be an index set and for every $t \in T$, A_t a subset of objects and B_t a subset of attributes. The concept lattice $C(I)$ is a complete lattice in which infimum and supremum are given by:

$$\bigwedge_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcup_{t \in T} B_t \right)^{\downarrow\uparrow} \right),$$

$$\bigvee_{t \in T} (A_t, B_t) = ((\bigcup_{t \in T} A_t)^{\uparrow\downarrow}, \bigcap_{t \in T} B_t).$$

A complete lattice $(L, \preceq)$ is isomorphic to $(C(I), \leq)$ if and only if there are functions $f : X \rightarrow V$ and $g : Y \rightarrow V$ such that $f(X)$ satisfies $v = \bigvee \{x \in f(X) \mid v \leq x\}$ and $g(Y)$ satisfies $v = \bigwedge \{y \in g(Y) \mid v \leq y\}$ for $v \in C(I)$, and $(x, y) \in I$ is equivalent to $fx \preceq gy$ for all $x \in X$ and $y \in Y$. $\square$

In general, concept lattices of different formal contexts are different. They have a case to be isomorphic. If x and x' are objects with $x \neq x'$ but $\{x\}^\uparrow = \{x'\}^\uparrow$, then according to the basic theorem a new formal context $(X \setminus \{x\}, Y, I \cap ((X \setminus \{x\}) \times Y))$ obtained by deleting x has a concept lattice isomorphic to (X, Y, I) .

$$(C(I), \leq) \cong (C(I \cap ((X \setminus \{x\}) \times Y)), \leq).$$

In Section 5, we will give a relational formulation for formal concepts and prove several theorems by employing relational calculus in Dedekind category.

3 Dedekind Categories

In this section, we recall the definition of a kind of relation category which we will call Dedekind categories following Olivier and Serrato [4]. Dedekind categories are equivalent to locally complete division allegories introduced in Freyd and Scedrov [3].

Throughout this paper, a morphism α from an object X into an object Y in a Dedekind category (which will be defined below) will be denoted by a half arrow, and the composite of a morphism $\alpha : X \rightarrow Y$ followed by a morphism $\beta : Y \rightarrow Z$ will be written as $\alpha\beta : X \rightarrow Z$. Also we will denote the identity morphism on X as id_X .

Definition 5. A Dedekind category $\mathcal{D}$ is a category satisfying the following:

D1. [Complete Heyting Algebra] For all pairs of objects X and Y the hom-set $\mathcal{D}(X, Y)$ consisting of all morphisms of X into Y is a complete Heyting algebra (namely, a complete distributive lattice) with the least morphism 0_{XY} and the greatest morphism ∇_{XY} . Its algebraic structure will be denoted by

$$\mathcal{D}(X, Y) = (\mathcal{D}(X, Y), \sqsubseteq, \sqcup, \sqcap, 0_{XY}, \nabla_{XY}).$$

D2. [Converse] There is given a converse operation $\sharp : \mathcal{D}(X, Y) \rightarrow \mathcal{D}(Y, X)$. That is, for all morphisms $\alpha, \alpha' : X \rightarrow Y$, $\beta : Y \rightarrow Z$, the following converse laws hold:

- (a) $(\alpha\beta)^\sharp = \beta^\sharp\alpha^\sharp$,
- (b) $(\alpha^\sharp)^\sharp = \alpha$,
- (c) $\alpha \sqsubseteq \alpha'$, then $\alpha^\sharp \sqsubseteq \alpha'^\sharp$.

D3. [Dedekind Formula] For all morphisms $\alpha : X \rightarrow Y$, $\beta : Y \rightarrow Z$ and $\gamma : X \rightarrow Z$, the Dedekind formula $\alpha\beta \sqcap \gamma \sqsubseteq \alpha(\beta \sqcap \alpha^\sharp\gamma)$ holds.

D4. [Residual Composition] For all morphisms $\alpha : X \rightarrow Y$ and $\beta : Y \rightarrow Z$, the residual composite $\alpha \ominus \beta : X \rightarrow Z$ is a morphism such that $\gamma \sqsubseteq \alpha \ominus \beta \iff \alpha^\# \gamma \sqsubseteq \beta$ for all morphisms $\gamma : X \rightarrow Z$. $\square$

For all morphisms α, β and γ , the following hold.

Proposition 1. *If $\alpha^\# \alpha \sqsubseteq \text{id}_Y$ then $\alpha(\beta \sqcap \gamma) = \alpha\beta \sqcap \alpha\gamma$.* $\square$

For a set of relation $\{\alpha_j : X \rightarrow Y | j \in J\}$, we define two relations $\sqcup_{j \in J} \alpha_j$ and $\sqcap_{j \in J} \alpha_j$ as follows:

$$\sqcup_{j \in J} \alpha_j = \{(a, b) \in X \times Y | \exists j \in J, (a, b) \in \alpha_j\},$$

$$\sqcap_{j \in J} \alpha_j = \{(a, b) \in X \times Y | \forall j \in J, (a, b) \in \alpha_j\}.$$

A morphism $f : X \rightarrow Y$ such that $f^\# f \sqsubseteq \text{id}_Y$ (*univalent*) and $\text{id}_X \sqsubseteq f f^\#$ (*total*) is called a *function* and may be introduced as $f : X \rightarrow Y$. In what follows, the word *relation* is a synonym for morphism of a Dedekind category.

A function $f : X \rightarrow Y$ is called a *surjection* if $f^\# f = \text{id}_Y$, and f is called a *injection* if $f f^\# = \text{id}_X$.

Next, we review some fundamental properties of residual composition [8].

Proposition 2. *Let $\alpha, \alpha' : A \rightarrow B$, $\beta, \beta' : B \rightarrow C$, $\gamma : C \rightarrow D$, $\mu : V \rightarrow B$ and $\rho, \rho', \rho_j : V \rightarrow A (j \in J)$ be relations in $\mathcal{D}$. Then the following hold:*

- (1) *If $\alpha' \sqsubseteq \alpha$ and $\beta \sqsubseteq \beta'$ then $\alpha \ominus \beta \sqsubseteq \alpha' \ominus \beta'$.*
- (2) *$\text{id}_A \sqsubseteq \alpha \ominus \alpha^\#$.*
- (3) *$\alpha \ominus (\beta \ominus \gamma) = \alpha\beta \ominus \gamma$.*
- (4) *$\alpha \ominus (\beta \ominus \gamma)^\# = [\beta \ominus (\alpha \ominus \gamma^\#)]^\#$.*
- (5) *If α is a function then $\alpha \ominus \beta = \alpha\beta$.*
- (6) *If α is a function then $\alpha(\beta \ominus \gamma) = \alpha\beta \ominus \gamma$ and $(\beta \ominus \gamma)\alpha^\# = \beta \ominus \gamma\alpha^\#$.*
- (7) *If α is a function then $\beta\alpha \ominus \gamma = \beta \ominus \alpha\gamma$.*
- (8) *If $\mu \sqsubseteq \rho \ominus \alpha$ then $\rho \sqsubseteq \mu \ominus \alpha^\#$.* (Galois connection)
- (9) *$\rho \sqsubseteq (\rho \ominus \alpha) \ominus \alpha^\#$.*
- (10) *$((\rho \ominus \alpha) \ominus \alpha^\#) \ominus \alpha = \rho \ominus \alpha$.*
- (11) *$(\sqcup_{j \in J} \rho_j) \ominus \alpha = \sqcap_{j \in J} (\rho_j \ominus \alpha)$.*
- (12) *If $\rho = (\rho \ominus \alpha) \ominus \alpha^\#$ then there exists a relation $\mu : V \rightarrow Y$ such that $\rho = \mu \ominus \alpha^\#$.*
- (13) *If α and α' are functions such that $\alpha \sqsubseteq \alpha'$, then $\alpha = \alpha'$.* $\square$

We define four relations $\max_\zeta(\rho)$, $\min_\zeta(\rho)$, $\sup_\zeta(\rho)$ and $\inf_\zeta(\rho) : V \rightarrow X$ for two relations $\rho : V \rightarrow X$ and $\zeta : X \rightarrow X$ as following:

maximum $\max_\zeta(\rho) = \rho \sqcap (\rho \ominus \zeta)$,

minimum $\min_\zeta(\rho) = \rho \sqcap (\rho \ominus \zeta^\#)$,

supremum $\sup_\zeta(\rho) = (\rho \ominus \zeta) \sqcap ((\rho \ominus \zeta) \ominus \zeta^\#)$,

infimum $\inf_\zeta(\rho) = (\rho \ominus \zeta^\#) \sqcap ((\rho \ominus \zeta^\#) \ominus \zeta)$.

The relations satisfy the following proposition.

Proposition 3. *Let $\alpha : X \rightarrow Y$ be a function, and $\beta : X \rightarrow Y$ and $\zeta : Y \rightarrow Y$ relations. Then the following holds.*

- (a) $\max_{\zeta}(\alpha\beta) = \alpha \max_{\zeta}(\beta)$ and $\min_{\zeta}(\alpha\beta) = \alpha \min_{\zeta}(\beta)$,
- (b) $\sup_{\zeta}(\alpha\beta) = \alpha \sup_{\zeta}(\beta)$ and $\inf_{\zeta}(\alpha\beta) = \alpha \inf_{\zeta}(\beta)$.

□

A relation $\zeta : X \rightarrow X$ is called an *order* if $\text{id}_X \sqsubseteq \zeta$ (reflexive), $\zeta\zeta \sqsubseteq \zeta$ (transitive) and $\zeta \sqcap \zeta^{\sharp} \sqsubseteq \text{id}_X$ (antisymmetric). A relation $\zeta : X \rightarrow X$ is *complete* if $\sup_{\zeta}(\rho)$ is a function for any relation $\rho : V \rightarrow X$.

In this paper, we assume that the Dedekind category $\mathcal{D}$ has subobjects, i.e., for any relation $u \sqsubseteq \text{id}_X$ there exists an injection $j : S \rightarrow X$ such that $u = j^{\sharp}j$. The Dedekind category $\mathcal{D}$ satisfies the following proposition.

Proposition 4. *For a function $f : X \rightarrow Y$ of $\mathcal{D}$ there exists a surjection $q : X \rightarrow Q$ and an injection $i : Q \rightarrow Y$ such that $f = qi$. This property is usually called a epi-mono factorization.*

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ q \downarrow & \nearrow i & \\ Q & & \end{array}$$

□

4 Membership Relations

In this section, we introduce a membership relation. And, for any relation α , we define functions $\wp(\alpha)$ and $\wp_*(\alpha)$. The function $\wp_*(\alpha)$ have a large part of definition of a formal concept in Section 5. In addition, we define the power order by using the membership relation. The power order has relations and the orders of power set.

Definition 6. We define a power object $\wp(Y)$ and a membership relation $\ni_Y : \wp(Y) \rightarrow Y$ of an object of Dedekind category.

- (a) $(\ni_Y \ominus \ni_Y^{\sharp}) \sqcap (\ni_Y \ominus \ni_Y^{\sharp})^{\sharp} \sqsubseteq \text{id}_{\wp(Y)}$,
- (b) for any relation $\alpha : X \rightarrow Y$ there exists a unique function $f : X \rightarrow \wp(Y)$ such that $f\ni_Y = \alpha$.

□

The unique function f with $f\ni_Y = \alpha$ will be denoted by a symbol $\alpha^{\textcircled{a}}$. In particular, the function $\text{id}_X^{\textcircled{a}} : X \rightarrow \wp(X)$ is called the *singleton-set function* on X .

We assume the Dedekind category has power object for each object.

For a relation $\alpha : X \rightarrow Y$, two functions $\wp(\alpha), \wp_*(\alpha) : \wp(X) \rightarrow \wp(Y)$ are defined by

$$\wp(\alpha) = (\ni_X \alpha)^{\textcircled{a}} \quad \text{and} \quad \wp_*(\alpha) = (\ni_X \ominus \alpha)^{\textcircled{a}},$$

$$\text{that is } \wp(\alpha)\ni_Y = \ni_X \alpha \quad \text{and} \quad \wp_*(\alpha)\ni_Y = \ni_X \ominus \alpha.$$

$$\begin{array}{ccc}
\wp(X) & \xrightarrow{\wp(\alpha)} & \wp(Y) \\
\wp_X \downarrow & & \downarrow \wp_Y \\
X & \xrightarrow{\alpha} & Y
\end{array}$$

We show the following proposition of membership relations and the above two functions.

Proposition 5. *Let $\alpha : X \rightarrow Y$, $\beta : Y \rightarrow Z$ and $\gamma : Z \rightarrow W$ be relations. Then the following holds.*

- (a) $\alpha^{\textcircled{a}} \wp(\beta) = (\alpha\beta)^{\textcircled{a}}$,
- (b) $\wp(\alpha)\wp(\beta) = \wp(\alpha\beta)$ and $\wp(\text{id}_X) = \text{id}_{\wp(X)}$,
- (c) $\wp(\wp_X)\wp_X = \wp_{\wp(X)}\wp_X$,
- (d) $\wp_*(\wp_X)\wp_X = \wp_{\wp(X)} \ominus \wp_X$,
- (e) $\text{id}_X^{\textcircled{a}} \wp(\alpha)\wp_Y = \text{id}_X^{\textcircled{a}} \wp_*(\alpha)\wp_Y = \alpha$,
- (f) $\wp_*(\alpha)\wp_*(\beta)\wp_Z = (\wp_X \ominus \alpha) \ominus \beta$,
- (g) $\wp_*(\alpha)\wp_*(\beta)\wp_*(\gamma)\wp_W = ((\wp_X \ominus \alpha) \ominus \beta) \ominus \gamma$,
- (h) $\wp_*(\alpha)\wp_*(\alpha^{\sharp})\wp_*(\alpha) = \wp_*(\alpha)$. □

A relation $\Xi_X : \wp(X) \rightarrow \wp(X)$ is defined by $\Xi_X = \wp_X \ominus \wp_X^{\sharp}$ and called the *power order* on $\wp(X)$.

The power order Ξ satisfies the following proposition.

Proposition 6. *Let $\alpha : X \rightarrow Y$, $\beta : Y \rightarrow Z$ and $\rho : V \rightarrow \wp(X)$ be relations.*

- (a) *The power order Ξ_X is an order on $\wp(X)$,*
- (b) $\Xi_X \sqcap \Xi_X^{\sharp} = \text{id}_{\wp(X)}$,
- (c) $\sup_{\Xi_X} (\wp_{\wp(X)}) = \wp(\wp_X)$, (join)
- (d) $\inf_{\Xi_X} (\wp_{\wp(X)}) = \wp_*(\wp_X)$, (meet)
- (e) $\sup_{\Xi_X} (\rho) = (\rho\wp_X)^{\textcircled{a}}$, (Ξ_X is complete)
- (f) $\sup_{\Xi_Y} (\rho\wp(\alpha)) = \sup_{\Xi_X} (\rho)\wp(\alpha)$, ($\wp(\alpha)$ is sup-continuous)
- (g) $\sup_{\Xi_Z} (\alpha\beta^{\textcircled{a}}) = (\alpha\beta)^{\textcircled{a}}$.

Proof

- (a) The reflexivity $\text{id}_{\wp(X)} \sqsubseteq \wp_X \ominus \wp_X^{\sharp} = \Xi_X$ is trivial. And the transitivity follows from

$$\Xi_X \Xi_X = (\wp_X \ominus \wp_X^{\sharp})(\wp_X \ominus \wp_X^{\sharp}) \sqsubseteq \wp_X \ominus \wp_X^{\sharp} = \Xi_X.$$

By the Definition 6, the antisymmetry follows from

$$\Xi_X \sqcap \Xi_X^{\sharp} = (\wp_X \ominus \wp_X^{\sharp}) \sqcap (\wp_X \ominus \wp_X^{\sharp})^{\sharp} \sqsubseteq \text{id}_{\wp(X)}.$$

- (b) By the proposition 1 (b), $\text{id}_{\wp(X)} \sqsubseteq \Xi_X$ and $\text{id}_{\wp(X)} \sqsubseteq \Xi_X^{\sharp}$. Therefore $\text{id}_{\wp(X)} \sqsubseteq \Xi_X \sqcap \Xi_X^{\sharp}$. In the above proof(a), we already proved $\Xi_X \sqcap \Xi_X^{\sharp} \sqsubseteq \text{id}_{\wp(X)}$. Hence $\Xi_X \sqcap \Xi_X^{\sharp} = \text{id}_{\wp(X)}$.

(c) We have

$$\begin{aligned}
 \exists_{\wp(X)} \ominus \Xi_X &= \exists_{\wp(X)} \ominus (\exists_X \ominus \exists_X^\#) \{ \text{Def. of } \Xi_X : \Xi_X = \exists_X \ominus \exists_X^\# \} \\
 &= \exists_{\wp(X)} \exists_X \ominus \exists_X^\# \{ \text{Prop. 2.3} \} \\
 &= \wp(\exists_X) \exists_X \ominus \exists_X^\# \{ \text{Prop. 5.4} \} \\
 &= \wp(\exists_X) (\exists_X \ominus \exists_X^\#) \{ \wp(\exists_X) : \text{function and Prop. 2.6} \} \\
 &= \wp(\exists_X) \Xi_X \{ \text{Def. of } \Xi_X \}
 \end{aligned}$$

and so

$$\begin{aligned}
 (\exists_{\wp(X)} \ominus \Xi_X) \ominus \Xi_X^\# &= \wp(\exists_X) \Xi_X \ominus \Xi_X^\# \\
 &= \wp(\exists_X) (\Xi_X \ominus \Xi_X^\#) \{ \wp(\exists_X) : \text{function} \} \\
 &= \wp(\exists_X) \Xi_X^\#. \{ \text{Def. of } \Xi_X \}
 \end{aligned}$$

Hence it hold that

$$\begin{aligned}
 \sup_{\Xi_X} (\exists_{\wp(X)}) &= (\exists_{\wp(X)} \ominus \Xi_X) \sqcap [(\exists_{\wp(X)} \ominus \Xi_X) \ominus \Xi_X^\#] \{ \text{Def. of sup} \} \\
 &= \wp(\exists_X) \Xi_X \sqcap \wp(\exists_X) \Xi_X^\# \\
 &= \wp(\exists_X) (\Xi_X \sqcap \Xi_X^\#) \{ \wp(\exists_X) : \text{function and Prop. 1} \} \\
 &= \wp(\exists_X). \{ \text{Prop. 6.5} \}
 \end{aligned}$$

(d) We have

$$\begin{aligned}
 \exists_{\wp(X)} \ominus \Xi_X^\# &= \exists_{\wp(X)} \ominus (\exists \exists_X \ominus \exists_X^\#)^\# \{ \text{Def. of } \Xi_X \} \\
 &= [\exists_X \ominus (\exists_{\wp(X)} \ominus \exists_X)^\#]^\# \{ \text{Prop. 2.4} \} \\
 &= [\exists_X \ominus (\wp_*(\exists_X) \exists_X)^\#]^\# \{ \text{Prop. 5.4} \} \\
 &= [(\exists_X \ominus \exists_X^\#) \wp_*(\exists_X)^\#]^\# \{ \wp_*(\exists_X) : \text{function} \} \\
 &= \wp_*(\exists_X) \Xi_X^\# \{ \text{Def. of } \Xi_X \}
 \end{aligned}$$

and so

$$\begin{aligned}
 (\exists_{\wp(X)} \ominus \Xi_X^\#) \ominus \Xi_X &= \wp_*(\exists_X) \Xi_X^\# \ominus \Xi_X \\
 &= \wp_*(\exists_X) (\Xi_X^\# \ominus \Xi_X) \{ \wp_*(\exists_X) : \text{function} \} \\
 &= \wp_*(\exists_X) \Xi_X. \{ \Xi_X^\# \ominus \Xi_X = \Xi_X \}
 \end{aligned}$$

Therefore

$$\begin{aligned}
 \inf_{\Xi_X} (\exists_{\wp(X)}) &= (\exists_{\wp(X)} \ominus \Xi_X^\#) \sqcap [(\exists_{\wp(X)} \ominus \Xi_X^\#) \ominus \Xi_X] \{ \text{Def. of inf} \} \\
 &= \wp_*(\exists_X) \Xi_X^\# \sqcap \wp_*(\exists_X) \Xi_X \\
 &= \wp_*(\exists_X) (\Xi_X^\# \sqcap \Xi_X) \{ \wp_*(\exists_X) : \text{function} \} \\
 &= \wp_*(\exists_X). \{ \text{Prop. 6.5} \}
 \end{aligned}$$

$$\begin{aligned}
 \text{(e) } \sup_{\Xi_X} (\rho) &= \sup_{\Xi_X} (\rho^\circledast \ominus_{\wp(X)}) \{ \text{Def. of } \rho^\circledast : \rho = \rho^\circledast \ominus_{\wp(X)} \} \\
 &= \rho^\circledast \sup_{\Xi_X} (\exists_{\wp(X)}) \{ \rho^\circledast : \text{function and Prop. 3} \} \\
 &= \rho^\circledast \wp(\exists_X) \{ \text{Prop. 6.5} \} \\
 &= (\rho \exists_X)^\circledast. \{ \text{Prop. 5.2} \}
 \end{aligned}$$

- (f) $\sup_{\Xi_Y}(\rho\wp(\alpha)) = (\rho\wp(\alpha)(\ni_Y))^{\textcircled{\alpha}} \{ \text{Prop. 6.6} \}$
 $= (\rho\ni_X\alpha)^{\textcircled{\alpha}} \{ \text{Def. of } \wp(\alpha) \}$
 $= (\rho\ni_X)^{\textcircled{\alpha}}\wp(\alpha) \{ \text{Prop. 5.6} \}$
 $= \sup_{\Xi_X}(\rho)\wp(\alpha). \{ \text{Prop. 5.6} \}$
- (g) The identity simply follows from

$$\begin{aligned} \sup_{\Xi_Z}(\alpha\beta^{\textcircled{\alpha}}) &= (\alpha\beta^{\textcircled{\alpha}}\ni_Z)^{\textcircled{\alpha}} \{ \text{Prop. 6.6} \} \\ &= (\alpha\beta)^{\textcircled{\alpha}}. \{ \beta^{\textcircled{\alpha}}\ni_Z = \beta \} \end{aligned}$$

□

5 Formal Concepts in Dedekind Categories

In this section, we discuss formal concepts and concept lattices in Dedekind categories and we prove basic properties of formal concepts by using relational calculus.

Let $\alpha : X \rightarrow Y$ be a relation in a Dedekind category. The relation α corresponds to a formal context in Section 1. We define two functions $F : \wp(X) \rightarrow \wp(Y)$ and $G : \wp(Y) \rightarrow \wp(X)$, which correspond to the function from a set of objects to intent $A \mapsto A^\uparrow$ and the function from a set of attributes to extent $B \mapsto B^\downarrow$. Moreover we define an object $C(\alpha)$ which means the set of all formal concepts of α .

Definition 7. Let $\alpha : X \rightarrow Y$ be a relation. Define two functions $F : \wp(X) \rightarrow \wp(Y)$ and $G : \wp(Y) \rightarrow \wp(X)$ by

$$F = \wp_*(\alpha) \quad \text{and} \quad G = \wp_*(\alpha^\#).$$

□

Definition 8. Let $\alpha : X \rightarrow Y$ be a relation in $\mathcal{D}$. We can choose an injection $j : C(\alpha) \rightarrow \wp(Y)$ such that $j^\#j = F^\#F$. (The injection j exists, since $\mathcal{D}$ has subobjects.)

□

Those relations defined above are illustrated by the following diagram:

$$\begin{array}{ccccc} & & C(\alpha) & & \\ & & \downarrow j & & \\ \wp(X) & \xrightarrow{F} & \wp(Y) & \xrightarrow{G} & \wp(X) \\ \ni_X \downarrow & & \downarrow \ni_Y & & \downarrow \ni_X \\ X & \xrightarrow{\alpha} & Y & \xrightarrow{\alpha^\#} & X \end{array}$$

Then a relation $Fj^\# : \wp(X) \rightarrow C(\alpha)$ is a function and $Fj^\#j = FF^\#F = F$.

Because a Dedekind category $\mathcal{D}$ has subobjects, the power order $\Xi_X = \ni_X \circ \ni_X^\# : \wp(X) \rightarrow \wp(X)$ is the order on $\wp(X)$.

We define functions used to formulate theorems of formal context and formal context.

Definition 9. Two functions $f : X \rightarrow C(\alpha)$ and $g : Y \rightarrow C(\alpha)$ are defined by

$$f = \text{id}_X^{\otimes} F j^{\sharp} \quad \text{and} \quad g = \text{id}_Y^{\otimes} G F j^{\sharp}.$$

□

We show the basic theorems of formal concept in Section 1.

Lemma 1

- (a) $F \Xi_Y^{\sharp} = \Xi_X G^{\sharp}$ (Galois connection).
- (b) $FG \sqsubseteq \Xi_X$, $GF \sqsubseteq \Xi_Y$, $FGF = F$ and $GFG = G$.
- (c) $\inf_{\Xi_Y} (\exists_{\wp(X)} F) = \sup_{\Xi_X} (\exists_{\wp(X)} F)$.
- (d) For all relations $\rho : V \rightarrow \wp(X)$, $\inf_{\Xi_Y} (\rho F) = \sup_{\Xi_X} (\rho F)$.

Proof

- (a) $\Xi_X G^{\sharp} = (\exists_X \odot \exists_X^{\sharp}) G^{\sharp} \quad \{ \text{Def. of } \Xi_X \}$
 $= \exists_X \odot (G \exists_X)^{\sharp} \quad \{ G : \text{function} \}$
 $= \exists_X \odot (\exists_Y \odot \alpha^{\sharp})^{\sharp} \quad \{ \text{Def. of } G \}$
 $= [\exists_Y \odot (\exists_X \odot \alpha)^{\sharp}]^{\sharp} \quad \{ \text{Prop. 2.4} \}$
 $= [\exists_Y \odot (F \exists_Y)^{\sharp}]^{\sharp} \quad \{ \text{Def. of } F \}$
 $= F(\exists_Y \odot \exists_Y^{\sharp})^{\sharp} \quad \{ F : \text{function} \}$
 $= F \Xi_Y^{\sharp}. \quad \{ \text{Def. of } \Xi_Y \}$
- (b) We prove $FG \sqsubseteq \Xi_X$

$$\begin{aligned} \Xi_X &\sqsubseteq \Xi_X FG(FG)^{\sharp} \quad \{ FG : \text{total} \} \\ &\sqsubseteq \Xi_X F \Xi_Y^{\sharp} G(FG)^{\sharp} \quad \{ \text{id}_Y \sqsubseteq \Xi_Y \} \\ &= \Xi_X \Xi_X G^{\sharp} G(FG)^{\sharp} \quad \{ \text{Lemma 1.1a} \} \\ &\sqsubseteq \Xi_X (FG)^{\sharp}. \quad \{ G : \text{univalent} \} \end{aligned}$$

Hence

$$\begin{aligned} \Xi_X FG &\sqsubseteq \Xi_X (FG)^{\sharp} FG \quad \{ \Xi_X \sqsubseteq \Xi_X (FG)^{\sharp} \} \\ &\sqsubseteq \Xi_X. \quad \{ FG : \text{total} \} \end{aligned}$$

On the other hands, we have $\text{id}_X FG \sqsubseteq \Xi_X FG$ by $\text{id}_{\wp(X)} \sqsubseteq \Xi_X$ and so $FG \sqsubseteq \Xi_X$. The proof of $GF \sqsubseteq \Xi_Y$ is similar. Also $FGF = F$ and $GFG = G$ follow from Proposition 3.3.

- (c) $\exists_{\wp(X)} F \odot \Xi_Y^{\sharp} = \exists_{\wp(X)} \odot F \Xi_Y^{\sharp} \quad \{ F : \text{function and Prop. 2.7} \}$
 $= \exists_{\wp(X)} \odot \Xi_X G^{\sharp} \quad \{ \text{Lemma 1.1a} \}$
 $= (\exists_{\wp(X)} \odot \Xi_X) G^{\sharp} \quad \{ G : \text{function} \}$
 $= \wp(\exists_X) \Xi_X G^{\sharp} \quad \{ \exists_{\wp(X)} \odot \Xi_X = \wp(\exists_X) \Xi_X \}$
 $= \sup_{\Xi_X} (\exists_{\wp(X)}) \Xi_X G^{\sharp} \quad \{ \text{Prop. 6.1c} \}$
 $= \sup_{\Xi_X} (\exists_{\wp(X)}) F \Xi_Y^{\sharp}. \quad \{ \text{Lemma 1.1a} \}$

Therefore

$$\begin{aligned} &\inf_{\Xi_Y} (\exists_{\wp(X)} F) \\ &= \max_{\Xi_Y} (\exists_{\wp(X)} F \odot \Xi_Y^{\sharp}) \quad \{ \text{Def. of max and inf} \} \\ &= \max_{\Xi_Y} (\sup_{\Xi_X} (\exists_{\wp(X)}) F \Xi_Y^{\sharp}) \\ &= \sup_{\Xi_X} (\exists_{\wp(X)}) F \max_{\Xi_Y} (\Xi_Y^{\sharp}) \quad \{ \sup_{\Xi_X} (\exists_{\wp(X)}) F : \text{function} \} \\ &= \sup_{\Xi_X} (\exists_{\wp(X)}) F. \quad \{ \max_{\Xi_Y} (\Xi_Y^{\sharp}) = \text{id}_Y \} \end{aligned}$$

(d) Take a unique function $k : V \rightarrow \wp\wp(X)$ such that $k \ni_{\wp(X)} = \rho$. Then we have

$$\begin{aligned}
 \inf_{\Xi_Y}(\rho F) &= \inf_{\Xi_Y}(k \ni_{\wp(X)} F) \quad \{ k \ni_{\wp(X)} = \rho \} \\
 &= k \inf_{\Xi_Y}(\ni_{\wp(X)} F) \quad \{ k : \text{function} \} \\
 &= k \sup_{\Xi_X}(\ni_{\wp(X)} F) \quad \{ \text{Lemma 1 (c)} \} \\
 &= \sup_{\Xi_X}(k \ni_{\wp(X)} F) \quad \{ k : \text{function} \} \\
 &= \sup_{\Xi_X}(\rho F). \quad \{ k \ni_{\wp(X)} = \rho \}
 \end{aligned}
 \quad \square$$

Lemma 1 (a) generalizes Theorem 1 (d), and Lemma 1 (b) corresponds to Theorem 1 (b) and (c). Next we show the completeness of $C(\alpha)$.

Theorem 3. Define an order $\xi_Y : C(\alpha) \rightarrow C(\alpha)$ by $\xi_Y = j \Xi_Y j^\sharp$. Then for all relations $\rho : V \rightarrow C(\alpha)$ the following holds:

- (a) $\inf_{\xi_Y}(\rho) = \sup_{\Xi_X}(\rho j F^\sharp) F j^\sharp$,
 (b) $\sup_{\xi_Y}(\rho) = \inf_{\Xi_X}(\rho j G) F j^\sharp$. (ξ_Y is complete)

Proof. Set $U_Y = \sup_{\Xi_Y}(\ni_{\wp(Y)}) = \wp(\ni_Y)$ and $M_Y = \inf_{\Xi_Y}(\ni_{\wp(Y)})$.

$$\begin{array}{ccccc}
 \wp(C(\alpha)) & \xrightarrow{\wp(j)} & \wp\wp(Y) & \xrightarrow{M} & \wp(Y) \\
 \downarrow \ni_{C(\alpha)} & & \downarrow \ni_{\wp(Y)} & & \downarrow \ni_Y \\
 C(\alpha) & \xrightarrow{j} & \wp(Y) & \xrightarrow{\ni_Y} & Y
 \end{array}$$

- (a) First we prove $\sup_{\Xi_X}(\rho j F^\sharp) F j^\sharp \sqsubseteq \inf_{\xi_Y}(\rho)$.

$$\begin{aligned}
 &\inf_{\xi_Y}(\rho) \\
 &= (\rho \odot \xi_Y^\sharp) \sqcap [(\rho \odot \xi_Y^\sharp) \odot \xi_Y] \quad \{ \text{Def. of inf} \} \\
 &= (\rho j \odot \Xi_Y^\sharp) j^\sharp \sqcap [(\rho j \odot \Xi_Y^\sharp) j^\sharp j \odot \Xi_Y] j^\sharp \quad \{ \text{Def. of } \xi_Y \} \\
 &\sqsupseteq (\rho j \odot \Xi_Y^\sharp) j^\sharp \sqcap [(\rho j \odot \Xi_Y^\sharp) \odot \Xi_Y] j^\sharp \quad \{ j^\sharp j \sqsubseteq \text{id}_{\wp(Y)} \} \\
 &= [(\rho j \odot \Xi_Y^\sharp) \sqcap (\rho j \odot \Xi_Y^\sharp) \odot \Xi_Y] j^\sharp \quad \{ j : \text{function} \} \\
 &= \inf_{\Xi_Y}(\rho j) j^\sharp \quad \{ \text{Def of inf} \} \\
 &= \inf_{\Xi_Y}(\rho j F^\sharp F) j^\sharp \quad \{ j = j j^\sharp j = j F^\sharp F \} \\
 &= \sup_{\Xi_X}(\rho j F^\sharp) F j^\sharp. \quad \{ \text{Lemma 1(a)} \}.
 \end{aligned}$$

Therefore we have $\inf_{\xi_Y}(\rho) = \sup_{\Xi_X}(\rho j F^\sharp) F j^\sharp$ because $\sup_{\Xi_X}(\rho j F^\sharp) F j^\sharp$ and $\inf_{\xi_Y}(\rho)$ are functions.

- (b) $\rho \odot \xi_Y = \rho \odot j \Xi_Y j^\sharp \quad \{ \text{Def. of } \xi_Y \}$
 $= (\rho \odot j \Xi_Y) j^\sharp \quad \{ j : \text{function} \}$
 $= (\rho \odot j \Xi_Y) F^\sharp F j^\sharp \quad \{ j^\sharp = j^\sharp j j^\sharp = F^\sharp F j^\sharp \}$
 $= (\rho \odot j \Xi_Y F^\sharp) F j^\sharp \quad \{ F : \text{function} \}$
 $= (\rho \odot j G \Xi_X^\sharp) F j^\sharp \quad \{ \text{Lemma 1(a)} \}$
 $= (\rho j G \odot \Xi_X^\sharp) F j^\sharp \quad \{ j G : \text{function} \}$

and

$$\begin{aligned}
 (\rho \odot \xi_Y) \odot \xi_Y^\# &= (\rho jG \odot \Xi_X^\#) Fj^\# \odot \xi_Y^\# \\
 &= (\rho jG \odot \Xi_X^\#) Fj^\# \odot j\Xi_Y^\# j^\# && \{ \text{Def. of } \xi_Y \} \\
 &= (\rho jG \odot \Xi_X^\#) \odot Fj^\# j\Xi_Y^\# j^\# && \{ Fj^\# : \text{function} \} \\
 &= (\rho jG \odot \Xi_X^\#) \odot F\Xi_Y^\# j^\# && \{ Fj^\# j = FF^\# F = F \} \\
 &= (\rho jG \odot \Xi_X^\#) \odot \Xi_X G^\# j^\# && \{ \text{Lemma } \boxed{11a} \} \\
 &= [(\rho jG \odot \Xi_X^\#) \odot \Xi_X G^\# F^\#] Fj^\# && \{ j^\# = F^\# Fj^\# \} \\
 &\sqsupseteq [(\rho jG \odot \Xi_X^\#) \odot \Xi_X] Fj^\#. && \{ \Xi_X \sqsubseteq \Xi_X G^\# F^\# \}
 \end{aligned}$$

Hence

$$\begin{aligned}
 \sup_{\xi_Y}(\rho) &= (\rho \odot \xi_Y) \sqcap [(\rho \odot \xi_Y) \odot \xi_Y^\#] && \{ \text{Def. of sup} \} \\
 &\sqsupseteq (\rho jG \odot \Xi_X^\#) Fj^\# \sqcap [(\rho jG \odot \Xi_X^\#) \odot \Xi_X] Fj^\# \\
 &\sqsupseteq ((\rho jG \odot \Xi_X^\#) \sqcap [(\rho jG \odot \Xi_X^\#) \odot \Xi_X]) Fj^\# && \{ (\alpha \odot \beta)\gamma \sqsubseteq \alpha\gamma \odot \beta\gamma \} \\
 &= \inf_{\Xi_X}(\rho jG) Fj^\#. && \{ \text{Def. of inf} \}
 \end{aligned}$$

Therefore we have $\sup_{\xi_Y}(\rho) = \inf_{\Xi_X}(\rho jG) Fj^\#$ because $\inf_{\Xi_X}(\rho jG) Fj^\#$ and $\sup_{\xi_Y}(\rho)$ are functions. $\square$

The following theorems mean that f is infimum-dense, g is supremum-dense and the formal context corresponding to formal concepts $C(\alpha)$ exists.

Theorem 4. *The functions f and g satisfy the following properties.*

- (a) $\xi_Y f^\# f \odot \xi_Y^\# = \xi_Y^\#$.
- (b) $\xi_Y^\# g^\# g \odot \xi_Y = \xi_Y$.
- (c) $f \xi_Y^\# g^\# = \alpha$.

Proof. (a) $\xi_Y f^\# f \odot \xi_Y^\# = j\Xi_Y j^\# f^\# f \odot j\Xi_Y^\# j^\#$ $\{ \text{Def. of } \xi_Y \}$

$$\begin{aligned}
 &= j[\Xi_Y(fj)^\# \odot fj\Xi_Y^\#] j^\# && \{ j, f : \text{function} \} \\
 &= j[\Xi_Y(\text{id}^\odot F)^\# \odot \text{id}^\odot F\Xi_Y^\#] j^\# && \{ fj = \text{id}_X^\odot F \} \\
 &= j[(\exists_Y \odot \alpha^\#) \odot (\exists_Y \odot \alpha^\#)^\#] j^\# && \{ \text{id}_X^\odot F \exists_Y = \alpha \} \\
 &= j[G \exists_X \odot (G \exists_X)^\#] j^\# && \{ \text{Def. of } G \} \\
 &= jG(\exists_X \odot \exists_X^\#) G^\# j^\# && \{ G : \text{function} \} \\
 &= jG\Xi_X G^\# j^\# && \{ \text{Def. of } \Xi_X \} \\
 &= jGF\Xi_Y^\# j^\# && \{ \text{Lemma } \boxed{11a} \} \\
 &= j\Xi_Y^\# j^\# && \{ jGF = j \} \\
 &= \xi_Y^\#. && \{ \text{Def. of } \xi_Y \}
 \end{aligned}$$

- (b) We first prove $g\xi_Y = \exists_Y^\# j^\#$.

$$\begin{aligned}
 g\xi_Y &= gj\Xi_Y j^\# && \{ \text{Def. of } \xi_Y \} \\
 &= gj\Xi_Y F^\# Fj^\# && \{ j^\# = F^\# Fj^\# \} \\
 &= gjG\Xi_X^\# Fj^\# && \{ \text{Lemma } \boxed{11a} \} \\
 &= \text{id}_Y^\odot GFj^\# jG\Xi_X^\# Fj^\# && \{ \text{Def. of } g \} \\
 &= \text{id}_Y^\odot GFG\Xi_X^\# Fj^\# && \{ Fj^\# j = F \} \\
 &= \text{id}_Y^\odot G\Xi_X^\# Fj^\# && \{ \text{Lemma } \boxed{11b} \} \\
 &= \text{id}_Y^\odot \Xi_Y F^\# Fj^\# && \{ \text{Lemma } \boxed{11a} \} \\
 &= \text{id}_Y^\odot \Xi_Y j^\# && \{ jF^\# F = j \} \\
 &= \exists_Y^\# j^\#. && \{ \text{id}_Y^\odot \exists_Y = \text{id}_Y \}
 \end{aligned}$$

Therefore we have

$$\begin{aligned}
 \xi_Y^\# g^\# g \odot \xi_Y &= \xi_Y^\# g^\# \odot g \xi_Y & \{ g : \text{function} \} \\
 &= j \ni_Y \odot \ni_Y^\# j^\# & \{ g \xi_Y = \ni_Y^\# j^\# \} \\
 &= j(\ni_Y \odot \ni_Y^\#) j^\# & \{ j : \text{function} \} \\
 &= j \Xi_Y j^\#. & \{ \text{Def. of } \Xi_Y \}
 \end{aligned}$$

(c) It holds that

$$\begin{aligned}
 g \xi_Y f^\# &= \ni_Y^\# j^\# f^\# & \{ g \xi_Y = \ni_Y^\# j^\# \} \\
 &= (f j \ni_Y)^\# \\
 &= (\text{id}_X^\oplus F j^\# j \ni_Y)^\# & \{ \text{Def. of } f \} \\
 &= (\text{id}_X^\oplus F \ni_Y)^\# & \{ F j^\# j = F \} \\
 &= (\text{id}_X^\oplus \ni_X \odot \alpha)^\# & \{ \text{Def. of } F \} \\
 &= \alpha^\#. & \{ \text{id}_X^\oplus \ni_X = \text{id}_X \}
 \end{aligned}$$

□

Next, we consider a construction of reduced contexts without same rows. In formal concept analysis, the construction is used to make formal contexts more simply.

Let $\alpha : X \rightarrow Y$ be a relation, $F : \wp(X) \rightarrow \wp(Y)$ a function and $j : C(\alpha) \rightarrow \wp(Y)$ an injection such that $j^\# j = F^\# F$. In addition for a relation $\gamma : Z \rightarrow X$, we construct a function $F' : \wp(Z) \rightarrow \wp(X)$ and an injection $j' : C(\gamma \odot \alpha) \rightarrow \wp(Y)$ with $F' = \wp_*(\gamma \odot \alpha)$ and $j'^\# j' = F'^\# F'$, respectively.

The above relations are illustrated by the following diagram:

$$\begin{array}{ccccc}
 & & C(\gamma \odot \alpha) & & C(\alpha) \\
 & & \searrow j' & & \downarrow j \\
 \wp(Z) & \xrightarrow{\wp(\gamma)} & \wp(X) & \xrightarrow{\wp_*(\alpha)} & \wp(Y) \\
 \ni_Z \downarrow & & \downarrow \ni_X & & \downarrow \ni_Y \\
 Z & \xrightarrow{\gamma} & X & \xrightarrow{\alpha} & Y
 \end{array}$$

The following lemma gives a relationship between $C(\gamma \odot \alpha)$ and $C(\alpha)$.

Lemma 2

- (a) $C(\gamma \odot \alpha)$ is a subobject of $C(\alpha)$.
- (b) If $\gamma^\# \gamma = \text{id}_X$ then $C(\gamma \odot \alpha)$ is isomorphic to $C(\alpha)$.

Proof (a) We have to show $j'^\# j' \sqsubseteq j^\# j$.

$$\begin{aligned}
 F' \ni_Y &= \wp_*(\gamma \odot \alpha) \ni_Y & \{ \text{Def. of } F' \} \\
 &= \ni_X \odot (\gamma \odot \alpha) & \{ \text{Def. of } \wp_*(\gamma \odot \alpha) \} \\
 &= \ni_X \gamma \odot \alpha & \{ \text{Prop. 2.3} \} \\
 &= \wp(\gamma) \ni_X \odot \alpha & \{ \text{Def. of } \wp(\gamma) \} \\
 &= \wp(\gamma) (\ni_X \odot \alpha) & \{ \wp(\gamma) : \text{function} \} \\
 &= \wp(\gamma) \wp_*(\alpha) \ni_Y & \{ \text{Def. of } \wp_*(\alpha) \} \\
 &= \wp(\gamma) F \ni_Y. & \{ \text{Def. of } F \}
 \end{aligned}$$

Hence we have $F' = \wp(\gamma)F$ by the extensionality of membership relations and

$$\begin{aligned}
 j'^{\sharp}j' &= F'^{\sharp}F' && \{ \text{Def. of } j' \} \\
 &= F'^{\sharp}\wp(\gamma)^{\sharp}\wp(\gamma)F \\
 &\sqsubseteq F'^{\sharp}F && \{ \wp(\gamma)^{\sharp}\wp(\gamma) \sqsubseteq \text{id}_{\wp(x)} \} \\
 &= j^{\sharp}j. && \{ \text{Def. of } j \}
 \end{aligned}$$

- (b) Assume $\gamma^{\sharp}\gamma = \text{id}_Y$. It is trivial that $\alpha = \gamma^{\sharp}\ominus(\gamma\ominus\alpha)$. Then, by Lemma.2.(a), $C(\alpha)$ is a subobject of $C(\gamma\ominus\alpha)$. Hence $C(\alpha)$ and $C(\gamma\ominus\alpha)$ are isomorphic. $\square$

Corollary 1. *Let $\alpha : X \rightarrow Y$ and $\beta : Q \rightarrow Y$ be relations. If $\alpha = q\beta$ for some surjection $q : X \rightarrow Q$, then $C(\alpha)$ and $C(\beta)$ are isomorphic.* $\square$

By Proposition 2, for any relation $\alpha : X \rightarrow Y$, the function $\alpha^{\textcircled{a}}$ can be decomposed into a surjection $q : X \rightarrow Q$ and an injection $i : Q \rightarrow \wp(Y)$. Then we have

$$\alpha = \alpha^{\textcircled{a}}\ni_Y = qi\ni_Y.$$

The relation $i\ni_Y$ is a reduced formal context of α . Corollary 1 indicates that concept lattices $C(\alpha)$ and $C(i\ni_Y)$ of an original formal context α and the reduced formal context $i\ni_Y$ are mutually isomorphic.

6 Summary and Outlook

In this paper, the frame and definition of formal concepts were formulated by use of relational calculus, and some basic theorems were proved. Moreover, our results might be applied to fuzzy and multivalued relations through this formulation. In the future, relation calculus might demonstrate more theorems of formal concepts. Really, it could completely represent an algorithm of the analytic method.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
2. Berghammer, R.: Computation of cut completions and concept lattices using relational algebra and relview. Journal on Relational Methods in Computer Science 1, 50–72 (2004)
3. Freyd, P.J., Scedrov, A.: Categories, Allegories. North-Holland, Amsterdam (1990)
4. Olivier, J.P., Serrato, D.: Catégories de dedekind. morphismes dans les catégories de schröder. C. R. Acad. Sci. Paris 260, 939–941 (1980)
5. Schmidt, G., Ströhlein, T.: Relations and graphs – Discrete Mathematics for Computer Science –. Springer, Heidelberg (1993)
6. Tarski, A.: On the calculus of relations. Journal of Symbolic Logic 6, 73–89 (1941)
7. Kawahara, Y.: Urysohn's lemma in schröder categories. Bull. Inform. Cybernet 39, 69 (2007)
8. Kawahara, Y.: Theory of relations. Lecture note (japanese)

The Structure of the One-Generated Free Domain Semiring

Peter Jipsen¹ and Georg Struth²

¹ Chapman University, One University Dr, Orange, CA 92866, USA
jipsen@chapman.edu

² The University of Sheffield, 211 Portobello Street, Sheffield S1 4DP, UK
G.Struth@dcs.shef.ac.uk

Abstract. This note gives an explicit construction of the one-generated free domain semiring. In particular it is proved that the elements can be represented uniquely by finite antichains in the poset of finite strictly decreasing sequences of nonnegative integers. It is also shown that this domain semiring can be represented by sets of binary relations with union, composition and relational domain as operations.

1 Introduction

A *semiring* is an algebra of the form $(A, +, 0, \cdot, 1)$ such that $(A, +, 0)$ is a commutative monoid, $(A, \cdot, 1)$ is a monoid, and $\cdot$ distributes over all finite joins from the left and right (i.e. $x(y+z) = xy+xz$, $(x+y)z = xz+yz$ and $x0 = 0x = 0$). A semiring is *idempotent* if $x+x = x$. In this case, $(A, +, 0)$ is a (join-)semilattice with 0 as bottom element, and $\cdot$ preserves the join-semilattice order (denoted by $\leq$) in both arguments. The variety of idempotent semirings is denoted by **IS**.

Let X be a set of variables (or generators). By distributivity, every term t in the signature of semirings can be written as a finite join of terms of the free monoid $X^* = \bigcup_{n \in \mathbb{N}} X^n$ with 1 as the empty sequence and $\cdot$ as concatenation. Hence the free idempotent semiring over X , denoted by $F_{\mathbf{IS}}(X)$, is isomorphic to the set $\mathcal{P}_{\text{fin}}(X^*)$ of all finite subsets of words over the generators, with $+$ given by union and $\cdot$ given by the complex product $U \cdot V = \{uv : u \in U, v \in V\}$.

Consequently, the equational theory of idempotent semirings is decidable. However, their quasiequational theory and their uniform word problem are undecidable: The uniform word problem for semigroups is known to be undecidable, and every semigroup S is a subreduct of its powerset semiring $\mathcal{P}(S)$.

In this note we consider *domain semirings*, which are idempotent semirings with an additional unary operation d that has the properties of a *domain operation*. Domain semirings have first been introduced in a two-sorted setting in which the domain operation maps arbitrary semiring elements to a special Boolean subalgebra [DMS06]. The reason is that arbitrary semiring elements are intended to model the actions of some program or transition system whereas the elements of the Boolean subalgebra model the states of that system. This approach has recently been generalised to a one-sorted setting [DS08] and we base our considerations on this simpler and more flexible approach.

Our aim is an explicit description of the one-generated free domain semiring $F_{\text{DS}}(x)$. To this end we first describe the one-generated free domain monoid $F_{\text{DM}}(x)$. We then show that these elements are the join irreducibles of $F_{\text{DS}}(x)$, and we describe how they are ordered. Finally we show that $F_{\text{DS}}(x)$ is isomorphic to the set of finite antichains in the poset of join irreducibles.

We conclude with the result that $F_{\text{DS}}(x)$ is representable by a concrete algebra of binary relations, with union, empty relation, composition, identity relation, and relational domain as operations.

Examples of domain semirings are, for instance, reducts of relation algebras (with $d(x) = (x; x^\sim) \wedge 1$), as well as reducts of Kleene algebras with domain.

Computationally meaningful models of domain semirings include the idempotent semirings of binary relations with domain defined in the standard way; the idempotent semirings formed by sets of traces of a program (which are alternating sequences of state and action symbols) with domain defined by starting states of traces; or the idempotent semirings formed by sets of paths in a graph with domain defined again by sets of starting states [DMS06].

Applications of domain semirings and Kleene algebras with domain have been intensively studied. First, domain models enabledness conditions for actions in programs and transition systems. Second, the domain operation can easily be extended into a modal diamond operator that acts on the underlying algebra of domain elements [MS06]. This links the algebraic approach with more traditional logics of programs such as dynamic, temporal and Hoare logics. Also some standard semantics of programs, including the weakest precondition and weakest liberal precondition semantics, can be modelled in this setting. Many concrete applications can be found in this and previous RelMiCS conference proceedings.

The free domain semiring is interesting in these applications since it identifies exactly those terms of domain semirings that have the same denotation in all domain semirings and because it allows the definition of efficient proof and decision procedures.

The domain axioms of domain semirings are the same as for relation algebras and for Kleene algebras with domain, and since both relation algebras and Kleene algebras have rich and complex (quasi)equational theories, we will independently study the simpler equational theory of domain semirings in this note. Even in this setting the n -generated free algebras appear to be fairly complicated, but at least we are able to handle the one-generated case.

2 Domain Semirings

A *domain monoid* is an algebra $(M, \cdot, 1, d)$ such that $(M, \cdot, 1)$ is a monoid and $d : M \rightarrow M$ is a function that satisfies

- (D1) $d(x)x = x$,
- (D2) $d(xd(y)) = d(xy)$,
- (D3) $d(d(x)y) = d(x)d(y)$, and
- (D4) $d(x)d(y) = d(y)d(x)$.

It follows that

$$\begin{aligned} d(1) &= 1 && [\text{take } x = 1 \text{ in (D1)}], \\ d(d(x)) &= d(x) && [\text{take } y = 1 \text{ in (D2)}], \text{ and} \\ d(x)d(x) &= d(x) && [\text{take } y = x \text{ in (D3)}]. \end{aligned}$$

Hence the set $d(M) = \{d(x) : x \in M\}$ forms a meet semilattice with 1 as top element.

A *domain semiring* is an algebra $(A, +, 0, \cdot, 1, d)$ such that $(A, +, 0, \cdot, 1)$ is a semiring, $(A, \cdot, 1, d)$ is a domain monoid, and the additional axioms

$$d(x + y) = d(x) + d(y), \quad d(0) = 0 \quad \text{and} \quad d(x) + 1 = 1$$

hold [DS08]. Multiplying the last axiom by x on both sides and applying (D1) shows that every domain semiring is an idempotent semiring. The varieties of domain monoids and domain semirings are denoted by DM and DS respectively.

We note that the definition of domain semiring used here is more general than the notion of $\hat{\delta}$ -semiring in [DMS06] since we do not require a test-subsemiring or a complementation operation on tests.

Note also that every monoid expands to a domain monoid by taking d to be the constant function 1. Likewise, for any idempotent semiring we can obtain a domain semiring by defining $d(x) = 1$ if $x \neq 0$ and $d(0) = 0$. Therefore the quasiequational theory of domain monoids and of domain semirings is undecidable.

Lemma 1

- (a) *In every domain semiring, the axioms (D3) and (D4) are implied by the remaining axioms.*
- (b) *For any domain semiring A , the set $d(A) = \{d(x) : x \in A\}$ forms a distributive lattice.*

Proof. (a) Since $d(x + y) = d(x) + d(y)$ and $d(x) + 1 = 1$, it follows that d is order-preserving and $d(x) \leq 1$. Hence we use (D1) to calculate

$$d(x)d(y) = d(d(x)d(y))d(x)d(y) \leq d(1d(y))d(x)1 = d(y)d(x),$$

proving (D4). For (D3), we proceed similarly, using (D1) and (D2):

$$\begin{aligned} d(x)d(y) &= d(d(x)d(y))d(x)d(y) \\ &= d(d(x)y)d(x)d(y) \\ &\leq d(d(x)y) \\ &= d(d(d(x)y))d(d(x)y) \\ &\leq d(x)d(y). \end{aligned}$$

- (b) Birkhoff [Bir67] showed that a semiring is a distributive lattice iff it satisfies $x + 1 = 1$ and $xx = x$. Note that $d(A)$ is a subsemiring of A , and these axioms hold in $d(A)$. □

Proofs of the previous lemma with an automated theorem prover (such as Prover9 [McC07]) can also be found in [DS08].

3 Reduced Terms and Normal Forms

As usual, we define $x^0 = 1$ and $x^{n+1} = x^n x$.

Lemma 2. *In a domain monoid, if $m \leq n$ then*

$$d(x^m)x^n = x^n \quad \text{and} \quad d(x^m)d(x^n) = d(x^n).$$

Proof. Assuming $m \leq n$, we write $x^n = x^m x^{n-m}$, and using (D1) we have

$$d(x^m)x^n = d(x^m)x^m x^{n-m} = x^m x^{n-m} = x^n.$$

Now (D3) implies $d(x^m)d(x^n) = d(d(x^m)x^n) = d(x^n)$. □

We now describe a normal form for the elements of $F_{\text{DM}}(x)$. For $i, j \geq 0$, a *basic term* is of the form $x^i d(x^j)$. A concatenation of n basic terms is thus of the form

$$x^{i_1} d(x^{j_1}) x^{i_2} d(x^{j_2}) \cdots x^{i_n} d(x^{j_n}).$$

Such a term is said to be *reduced* if $j_k > i_{k+1} + j_{k+1}$ and $i_{k+1} > 0$ for all $k \in \{1, 2, \dots, n-1\}$. In particular, all basic terms are reduced.

Next we show that the domain of a reduced term is easy to determine. Together with the subsequent lemma, it follows that any term in the one-generated free domain semiring is equivalent to a term that has no nested occurrences of the domain symbol.

Lemma 3. *Let $t = x^{i_1} d(x^{j_1}) x^{i_2} d(x^{j_2}) \cdots x^{i_n} d(x^{j_n})$ be a reduced term. Then*

$$d(t) = d(x^{i_1+j_1}).$$

Proof. We use induction on n . For $n = 1$, the result follows from (D2). Suppose it holds for $n - 1$, and let $s = x^{i_1} d(x^{j_1}) \cdots d(x^{j_{n-2}}) x^{i_{n-1}}$. Using (D2) twice we have

$$d(t) = d((sd(x^{j_{n-1}})x^{i_n})d(x^{j_n})) = d(sd(x^{j_{n-1}})x^{i_n+j_n}) = d(sd(d(x^{j_{n-1}})x^{i_n+j_n}))$$

and since $j_{n-1} > i_n + j_n$ we obtain $d(t) = d(sd(x^{j_{n-1}}))$ from (D3) and the preceding lemma. By the inductive hypothesis, the last term is just $d(x^{i_1+j_1})$, as required. □

The concatenation of two reduced terms need not be reduced, but the next lemma shows how to rewrite any such product to reduced form.

Lemma 4. *In any domain monoid the following identities hold:*

- (a) $d(xy)xd(yz) = xd(yz)$,
- (b) if $0 \leq i \leq j + k$ then $d(x^i)x^j d(x^k) = x^j d(x^k)$.

Proof. (a) First we note that (D3) and (D1) yield

$$d(y)d(yz) = d(d(y)yz) = d(yz)$$

Using (D2) and (D1) we then obtain

$$d(xy)xd(yz) = d(xd(y))xd(y)d(yz) = xd(y)d(yz) = xd(yz).$$

(b) If $i \leq j$ then the result follows from Lemma 2. So suppose $i > j$ and $i \leq j+k$. Then $i-j \leq k$, hence, by the result in (a),

$$d(x^i)x^jd(x^k) = d(x^jx^{i-j})x^jd(x^{i-j}x^{k-(i-j)}) = x^jd(x^{i-j}x^{k-(i-j)}). \quad \square$$

Let $t = x^{i_1}d(x^{j_1})x^{i_2}d(x^{j_2}) \cdots x^{i_n}d(x^{j_n})$ be a concatenation of basic terms. The *x-length* of t is defined to be $\sum_{k=1}^n i_k$. Terms with zero *x-length* are of the form $d(x^i)$, and they are called *domain terms*. Part (b) of the preceding lemma can be used to eliminate redundant domain terms in any concatenation of basic terms, and this is repeated until the term is in reduced normal form. This process is obviously terminating and it is not hard to see that it has the Church-Rosser property, that is, it produces the same normal form regardless of the order in which domain terms are eliminated. Note also that rewriting terms to normal form preserves the *x-length*.

The reduced normal form described above, though rather compact, is not convenient for describing the partial order on the elements of the free domain monoid. On elements of the form $d(x^j)$, the order is induced by the meet-semilattice structure: $d(x^j) \leq d(x^k)$ iff $j \geq k$, hence these elements form a chain (see Fig. 1).

For concatenations of basic terms, we rewrite them in *expanded normal form*:

$$d(x^{j_0})xd(x^{j_1})xd(x^{j_2})x \cdots xd(x^{j_m}).$$

where each of the j_k are chosen to be as large as possible. This is justified by using part (b) of the preceding lemma in the reverse direction and with $j = 1$. For brevity we denote such a term by the sequence $(j_0, j_1, j_2, \dots, j_m)$ and note that this is always a strictly decreasing sequence of nonnegative integers. Let $\mathbf{P} = (P, \leq)$ be the set of all such sequences, ordered by reverse pointwise order. Thus sequences of different length are not comparable, and the maximal elements of this poset are

$$(0), \quad (1, 0), \quad (2, 1, 0), \quad \dots$$

corresponding to the terms

$$d(1) = 1, \quad d(x)xd(1) = x, \quad d(x^2)xd(x)xd(1) = x^2, \quad \dots$$

A diagram of an initial part of $\mathbf{P}$ is shown in Figures 1 and 2. A multiplication is defined on P by the following “ripple product”

$$(j_0, j_1, j_2, \dots, j_m) \cdot (k_0, k_1, k_2, \dots, k_n) = (j'_0, j'_1, j'_2, \dots, j'_m, k_1, k_2, \dots, k_n)$$

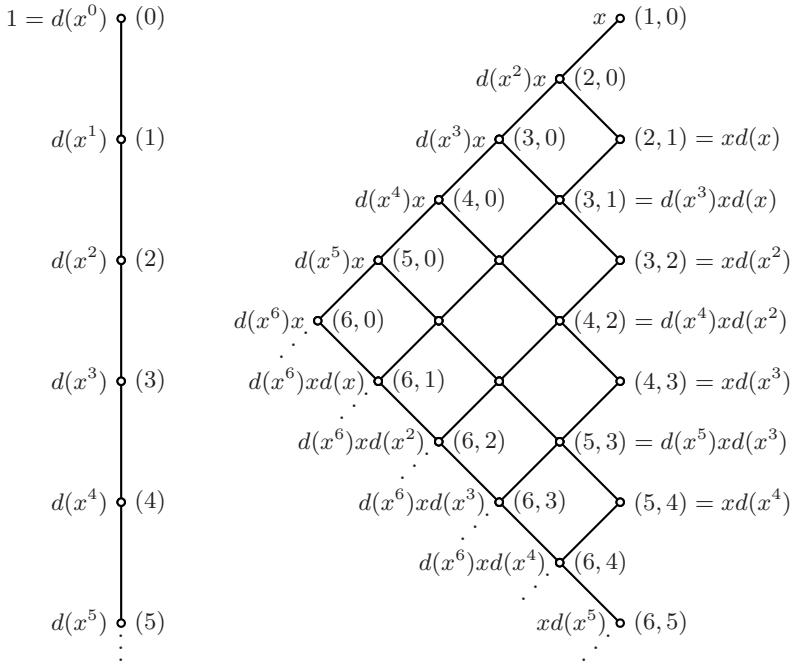


Fig. 1. Below 1 and x in the poset of join-irreducibles of $F_{\text{DS}}(x)$

where $j'_m = \max(j_m, k_0)$ and $j'_i = \max(j_i, j'_{i+1} + 1)$ for $i = m - 1, \dots, 2, 1, 0$. For example, $(7, 3, 2) \cdot (4, 3, 1) = (7, 5, 4, 3, 1)$, while $(4, 3, 1) \cdot (7, 3, 2) = (9, 8, 7, 3, 2)$. The motivation for this definition comes from observing that this is the result if we multiply the corresponding expanded normal forms and rewrite the product again in expanded normal form. It is tedious but not difficult to check that this operation is associative.

The domain of a sequence $(j_0, j_1, j_2, \dots, j_m)$ is the length-one sequence (j_0) , which corresponds to the domain term $d(x^{j_0})$.

Let $A(\mathbf{P})$ be the set of finite antichains of $\mathbf{P}$. A partial order is defined on $A(\mathbf{P})$ by $a \leq b$ iff $\downarrow a \subseteq \downarrow b$.

The multiplication is extended to antichains by using the complex product (i.e. $U \cdot V = \{uv : u \in U, v \in V\}$) and by removing all non-maximal elements.

4 Two Representation Theorems

We can now prove the main results of this note and show that the one-generated free domain semiring can be represented either in terms of antichains of integer sequences or in terms of sets of binary relations.

Theorem 1. *The join irreducibles of $F_{\text{DS}}(x)$ form a poset that is isomorphic to $\mathbf{P}$, and $F_{\text{DS}}(x)$ is isomorphic to $A(\mathbf{P})$.*

Proof. By distributivity, each domain semiring term $t(x)$ can be written as a finite join of expanded normal form terms. Hence any join irreducible element of $F_{\text{DS}}(x)$ can be represented by an expanded normal form term.

To show that $\mathbf{P}$ is the poset of these join irreducible, it suffices to show that all expanded normal forms are join irreducible, and that two expanded normal form terms can be distinguished in some domain monoid. We use a domain monoid of relations for the second part. Let $\mathbf{j} = (j_0, \dots, j_m)$ be a decreasing sequence of natural numbers, and define a relation $X_{\mathbf{j}}$ on $\mathbb{N} \times \mathbb{N}$ by $(u, v)X_{\mathbf{j}}(u', v')$ iff

$$(u = u' \text{ and } v + 1 = v' \leq j_u) \text{ or } (v = v' = 0 \text{ and } u + 1 = u' \leq m)$$

Let $t_{\mathbf{j}}(x)$ be the term that corresponds to the sequence $\mathbf{j}$. Then it is not hard to see that $((0, 0), (m, 0)) \in t_{\mathbf{j}}(X_{\mathbf{j}})$ but for any term s that is not above $t_{\mathbf{j}}$ in $\mathbf{P}$, $((0, 0), (m, 0)) \notin s(X_{\mathbf{j}})$ (see Fig. 3 for an illustration of $X_{\mathbf{j}}$).

To prove that the expanded normal form terms are join irreducible, it suffices to show that each such term t is not the join of the elements $s_1, \dots, s_k$ immediately below it in $\mathbf{P}$. For this result we consider the relation X that is the union of all the relations $X_{\mathbf{j}}$ (defined on disjoint base sets), where $\mathbf{j}$ ranges over the sequences that correspond to the terms $t, s_1, \dots, s_k$. If we evaluate t and $s_1 + \dots + s_k$ at this relation X , we see that t is strictly bigger, since it contains a pair from the base of its corresponding relation, which is not contained in $s_i(X)$ for any $i = 1, \dots, k$. $\square$

We now consider the question of representing domain semirings by algebras of binary relations. We first note that for free idempotent semirings this is always possible [BS78]. For a set X of generators, a concrete construction can be obtained by considering the complex algebra of the free group $F_{\text{Grp}}(X)$. This is always a representable relation algebra, with the elements of the group as disjoint relations. Since the free monoid X^* is a subset of the free group, the finite unions of the relations corresponding to singleton words give a relational representation of the free idempotent semiring with X as set of generators.

However, not all idempotent semirings can be represented by $\cup, \circ$ semirings of relations. In fact [And88, And91] showed that the class of algebras of relations, closed under $\cup, \circ$, though definable by quasiequations, is not finitely axiomatisable, hence it is strictly smaller than the finitely based variety of idempotent semirings. Similarly the class of algebras of relations closed under $\cup, \emptyset, \circ, \text{id}, d$, where $d(R) = R;R^{\sim} \cap \text{id}$, is a non-finitely axiomatisable quasivariety, but not a variety.

Theorem 2. *The one-generated free domain semiring can be represented by a domain semiring of binary relations.*

Proof. To see that $F_{\text{DS}}(x)$ can be represented by a collection of binary relations, with operations of union, composition and domain, it suffices to construct a relation X on a set U such that $s(X) \neq t(X)$ in the relation domain semiring $\mathcal{P}(U \times U)$ for any distinct pair of elements of $F_{\text{DS}}(x)$. This is done similarly to the proof of the preceding theorem, by taking X to be the union (over disjoint base sets) of all the relations $X_{\mathbf{j}}$ corresponding to the sequences $\mathbf{j} \in \mathbf{P}$.

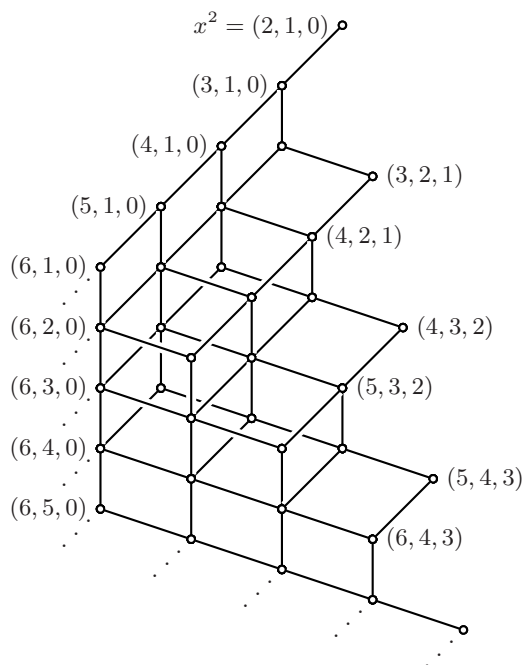


Fig. 2. Below x^2 in the poset of join-irreducibles of $F_{DS}(x)$

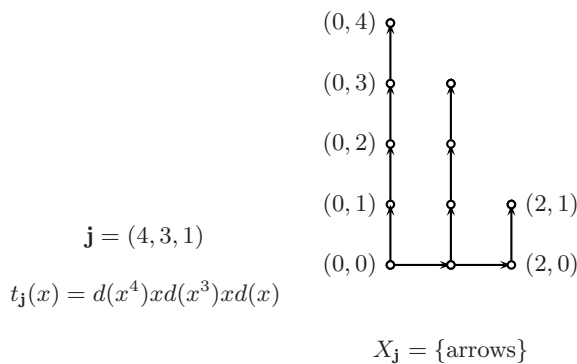


Fig. 3. The term and relation for $\mathbf{j} = (4, 3, 1)$

If s, t are distinct elements of the free domain semiring, then there exists a join irreducible that is below one of them, say s , but not below t . Let $\mathbf{j}$ be the decreasing sequence that corresponds to the expanded normal form $t_{\mathbf{j}}$ for this join irreducible element. Then $t_{\mathbf{j}}(X_{\mathbf{j}}) \subseteq s(X)$ but there is at least one ordered pair in $t_{\mathbf{j}}(X_{\mathbf{j}})$ that is not contained in $t(X)$, hence $s(X)$ and $t(X)$ are distinct relations. $\square$

5 Conclusion

So far our analysis has considered only the one-generated free domain semiring. Even the two-generated case is significantly more complex, since the description of the join irreducible elements is not so transparent (e.g. a term like $d(xd(y)x)$ does not appear to be equivalent to a concatenation of basic terms).

Future research is also aiming to describe the structure of free domain semirings in the presence of additional axioms. It has been shown in [DS08] that the domain algebras $d(S)$ induced by the domain axioms can be turned into (co-)Heyting algebras or Boolean algebras by imposing further constraints. In particular, adding the three axioms

$$a(x)x = 0, \quad a(xy) \leq a(xa(a(y))) \quad \text{and} \quad a(a(x)) + a(x) = 1$$

for an *antidomain* function $a : S \rightarrow S$ to the semiring axioms and defining domain as $d(x) = a(a(x))$ suffices to enforce that $d(S)$ is a Boolean algebra and to recover all theorems of the original two-sorted axiomatisation [DMS06]. Based on these results, in particular the structure of the free Boolean domain semirings certainly deserve further investigation.

References

- [And88] Andréka, H.: On the representation problem of distributive semilattice-ordered semigroups. Abstracts of the AMS 10(2), 174 (preprint, 1988)
- [And91] Andréka, H.: Representations of distributive lattice-ordered semigroups with binary relations. Algebra Universalis 28, 12–25 (1991)
- [Bir67] Birkhoff, G.: Lattice Theory, 3rd edn., vol. 25, pp. viii+420. AMS Colloquium Publications, AMS (1967)
- [BS78] Bredihin, D.A., Schein, B.M.: Representations of ordered semigroups and lattices by binary relations. Colloq. Math. 39, 1–12 (1978)
- [DMS06] Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM Transactions on Computational Logic 27(4), 798–833 (2006)
- [DS08] Desharnais, J., Struth, G.: Modal Semirings Revisited, Research Report CS-08-01, Department of Computer Science, The University of Sheffield (2008)
- [McC07] McCune, W.: Prover9 (2007), <http://www.prover9.org>
- [MS06] Möller, B., Struth, G.: Algebras of modal operators and partial correctness. Theoretical Computer Science 351, 221–239 (2006)

Determinisation of Relational Substitutions in Ordered Categories with Domain*

Wolfram Kahl

McMaster University, Hamilton, Ontario, Canada
kahl@cas.mcmaster.ca

Abstract. We present two different relational generalisations of substitutions, show that they both produce locally ordered categories with domain, and then develop the single-morphism “determiniser” concept that relies only on this framework, while still corresponding to conventional two-morphism unification in both examples. Central to this development is the determinacy concept of “characterisation by domain” introduced by Desharnais and Möller for Kleene algebras with domain; this is here applied in the weakest possible setting.

1 Introduction

Substitutions have been considered in a categorical context since Lawvere’s seminal work [14]. In that context, a unification problem can be stated as a pair of parallel arrows, and their most general unifier is then just their co-equaliser.

Relational substitution concepts allow more liberal ways to formulate unification problems, in particular as a *single*, relational morphism. In this paper, we consider two different categories for “relational” concepts of substitutions:

- “Relational substitutions” can be understood as non-deterministic variable bindings, and have a composition that corresponds to call-by-name, or “runtime choice”.
- “Substitution sets” correspond to non-deterministic choices of standard substitutions, and therefore more closely correspond to call-by-value, or “call-time choice”.

The greatest common denominator of these two relational substitution concepts is the setting of ordered categories with domain. In this setting, the essence of being “unified” can be captured via the determinacy concept of domain minimality, introduced by Desharnais and Möller [3]. We therefore replace the co-equaliser-based definition of unifiers with a new definition of “determiniser”, and show that this relates usefully to relational translations of conventional substitution problems in both relational substitution concepts.

In sections 2 and 4, we fix terminology and notation for categorical and syntactical issues respectively. “Relational substitutions” emerge as Kleisli category over a relator-based monad; we collect the necessary definitions in Sect. 3 and

* This research is supported by NSERC (National Science and Engineering Research Council of Canada).

show relevant properties of the Kleisli category. Then, in sections 5 and 6 we introduce the two relational substitution concepts and fit them into our categorical framework. Section 7 develops the determiniser concept as a unification concept that fits both substitution categories. Finally, in Sect. 8, we discuss some related work.

2 Ordered Categories with Domain

In our use of category theoretical concepts, we write composition using the “diagrammatic” convention:

Notation 2.1. In a *category*, we write $\mathbb{I}_{\mathcal{A}}$ for the identity on object $\mathcal{A}$, and $f : \mathcal{A} \rightarrow \mathcal{B}$ to say that f is a morphism from $\mathcal{A}$ to $\mathcal{B}$. The *homset* of all morphisms from $\mathcal{A}$ to $\mathcal{B}$ is also written $\text{Hom}(\mathcal{A}, \mathcal{B})$.

For two morphisms $f : \mathcal{A} \rightarrow \mathcal{B}$ and $g : \mathcal{B} \rightarrow \mathcal{C}$, we write $f ; g$ for their composition (and we have $(f ; g) : \mathcal{A} \rightarrow \mathcal{C}$). $\square$

Definition 2.2. An (*locally*) *ordered category* is a category in which on each homset $\text{Hom}(\mathcal{A}, \mathcal{B})$, there is an ordering $\subseteq_{\mathcal{A}, \mathcal{B}}$, and composition is monotonic in both arguments. $\square$

We will normally omit the subscripts, as they can be deduced from the context.

An endomorphism $p : \mathcal{A} \rightarrow \mathcal{A}$ is called a *subidentity* iff $p \subseteq \mathbb{I}_{\mathcal{A}}$.

We use the domain definition of [4], adapted to the setting of ordered categories:

Definition 2.3. An *ordered category with predomain* is an ordered category where for every morphism $R : \mathcal{A} \rightarrow \mathcal{B}$ there is a subidentity $\text{dom } R : \mathcal{A} \rightarrow \mathcal{A}$ such that for every subidentity $q : \mathcal{A} \rightarrow \mathcal{A}$, we have $q ; R \supseteq R$ iff $q \supseteq \text{dom } R$.

In an *ordered category with domain*, additionally the following “locality” condition holds:

$$\text{dom } (R ; S) = \text{dom } (R ; \text{dom } S)$$

Range $\text{ran } R$ is defined dually. $\square$

In allegory and relation algebra contexts, many properties are normally defined using converse; some of these can be defined using domain instead:

Definition 2.4. In an ordered category with domain (respectively range), we call a morphism $R : \mathcal{A} \rightarrow \mathcal{B}$

- *total* iff $\text{dom } R = \mathbb{I}_{\mathcal{A}}$,
- *surjective* iff $\text{ran } R = \mathbb{I}_{\mathcal{B}}$.

$\square$

For the property of univalence, defined using converse as $R^{\sim} ; R \subseteq \mathbb{I}$, it is harder to find an appropriate replacement that does not use converse; Desharnais and Möller have studied this problem extensively in [3]; we will mainly use the property they introduced as “characterisation by domain (CD)”:

Definition 2.5. In an ordered category with domain, a morphism $F : \mathcal{A} \rightarrow \mathcal{B}$ is called *deterministic* iff F is *domain-minimal*, i.e., iff

$$\forall R : \mathcal{A} \rightarrow \mathcal{B} \bullet \quad R \subseteq F \quad \Rightarrow \quad R = \text{dom } R ; F . \quad \square$$

For special cases of the local ordering we recall (e.g. from [11]):

Definition 2.6. An ordered category is called

- a *lower semilattice category* if each homset has binary meets,
- a *upper semilattice category* if each homset has binary joins, and composition distributes over these,
- a *complete upper semilattice category* if each homset has arbitrary joins, and composition distributes over these,
- *having zero morphisms*, if each homset has a least element (which is the join of the empty set), and these behave as zeros (which is distribution over the empty join).

A *Kleene category* is an upper semilattice category with zero morphisms where on homsets of endomorphisms there is an additional unary operation $_*$ such that $R^* = \mathbb{I}_A \cup R \cup R^* ; R^*$ and the induction laws hold:

$$Q ; R \subseteq Q \Rightarrow Q ; R^* \subseteq Q \quad \text{and} \quad R ; S \subseteq S \Rightarrow R^* ; S \subseteq S \quad \square$$

A complete upper semilattice category is automatically a Kleene category.

3 Ordered Monads

Functor and monad concepts are easily transferred to the setting of ordered categories — relators as “relational functors” have originally been introduced by Kawahara [12]; the following definitions are adapted to our setting from Backhouse [1]:

Definition 3.1. A *relator* between two ordered categories is a monotonic functor.

A *natural simulation* τ from a relator $\mathcal{F} : \mathbf{C} \rightarrow \mathbf{D}$ to a relator $\mathcal{G} : \mathbf{C} \rightarrow \mathbf{D}$ is a family of total and deterministic morphisms in $\mathbf{D}$ (which therefore needs domain) indexed with objects of $\mathbf{C}$ such that $\tau_{\mathcal{A}} : \mathcal{F} \mathcal{A} \rightarrow \mathcal{G} \mathcal{A}$ and for every $R : \mathcal{A} \rightarrow \mathcal{B}$ we have $\mathcal{F} R ; \tau_{\mathcal{B}} = \tau_{\mathcal{A}} ; \mathcal{G} R$.

An *ordered monad* is a triple $(\mathcal{M}, \eta, \mu)$ such that $\mathcal{M} : \mathbf{C} \rightarrow \mathbf{C}$ is an endo-relator, and $\eta : \mathbb{I} \rightarrow \mathcal{M}$ and $\mu : \mathcal{M} ; \mathcal{M} \rightarrow \mathcal{M}$ are natural simulations satisfying associativity: $\mu_{\mathcal{F} \mathcal{A}} ; \mu_{\mathcal{A}} = \mathcal{F} \mu_{\mathcal{A}} ; \mu_{\mathcal{A}}$ and the unit laws: $\eta_{\mathcal{M} \mathcal{A}} ; \mu_{\mathcal{A}} = \mathbb{I}_{\mathcal{M} \mathcal{A}}$ and $\mathcal{M} \eta_{\mathcal{A}} ; \mu_{\mathcal{A}} = \mathbb{I}_{\mathcal{M} \mathcal{A}}$. $\square$

For such an ordered monad over $\mathbf{C}$, the Kleisli category $\mathbb{K} \mathcal{M}$ is defined as usual, with η for identities, and composition of $R : \mathcal{A} \rightarrow \mathcal{M} \mathcal{B}$ and $S : \mathcal{B} \rightarrow \mathcal{M} \mathcal{C}$ defined as $R \circ S := R ; \mathcal{M} S ; \mu_{\mathcal{C}}$.

Monotonicity of composition in the Kleisli category (which inherits the ordering from $\mathbf{C}$) follows from monotonicity of composition in $\mathbf{C}$ together with monotonicity of $\mathcal{M}$.

Since $\eta_{\mathcal{A}}$ is deterministic, if $R \subseteq \eta_{\mathcal{A}}$ then $R = \text{dom } R ; \eta_{\mathcal{A}}$, so the subidentities in the Kleisli category are in one-to-one correspondence with the subidentities in $\mathbf{C}$, and domain is preserved:

Lemma 3.2. The Kleisli category for an ordered monad $(\mathcal{M}, \eta, \mu)$ over an ordered category $\mathbf{C}$ with domain is an ordered category with domain again, and $\text{dom}_{\mathbb{K}\mathcal{M}} R = (\text{dom}_{\mathbf{C}} R) ; \eta$. Furthermore, $S : \mathcal{A} \rightarrow \mathcal{M}\mathcal{B}$ is domain-minimal in $\mathbb{K}\mathcal{M}$ iff it is domain-minimal in $\mathbf{C}$.

PROOF: We show the last statement using the domain equation:

$$\begin{aligned}
 & S \text{ is domain-minimal in } \mathbb{K}\mathcal{M} \\
 \Leftrightarrow & \quad \forall R : \mathcal{A} \rightarrow \mathcal{M}\mathcal{B} \bullet R \subseteq S \Rightarrow R = \text{dom}_{\mathbb{K}\mathcal{M}} R ; S \\
 \Leftrightarrow & \quad \forall R : \mathcal{A} \rightarrow \mathcal{M}\mathcal{B} \bullet R \subseteq \sigma \Rightarrow R = \text{dom}_{\mathbf{C}} R ; \eta_{\mathcal{A}} ; \mathcal{M} S ; \mu_{\mathcal{B}} \\
 \Leftrightarrow & \quad \forall R : \mathcal{A} \rightarrow \mathcal{M}\mathcal{B} \bullet R \subseteq S \Rightarrow R = \text{dom}_{\mathbf{C}} R ; S ; \eta_{\mathcal{B}} ; \mu_{\mathcal{B}} \\
 \Leftrightarrow & \quad \forall R : \mathcal{A} \rightarrow \mathcal{M}\mathcal{B} \bullet R \subseteq S \Rightarrow R = \text{dom}_{\mathbf{C}} R ; S \\
 \Leftrightarrow & \quad \sigma \text{ is domain-minimal in } \mathbf{C} \quad \square
 \end{aligned}$$

From the definition of composition in the Kleisli category we easily obtain one half of join preservation:

Lemma 3.3. If $\mathbf{C}$ is a (complete) upper semilattice category (with zero morphisms), then composition in the Kleisli category distributes over binary (and arbitrary) (and empty) joins to its left.

PROOF: With join-distributivity in $\mathbf{C}$, we have (for a two-element, respectively arbitrary, respectively empty set S)¹

$$(\bigcup S) ; T = (\bigcup S) ; \mathcal{F} T ; \mu_{\mathcal{C}} = \bigcup \{S : S \bullet S ; \mathcal{F} T ; \mu_{\mathcal{C}}\} = \bigcup \{S : S \bullet S ; T\} \quad \square$$

Preservation of different kinds of joins in the right argument of composition additionally requires preservation of these joins by the relator $\mathcal{M}$:

Lemma 3.4. If $\mathbf{C}$ is a (complete) upper semilattice category (with zero morphisms) and the monad functor $\mathcal{M}$ preserves binary (and arbitrary) (and empty) joins, then the Kleisli category is a (complete) upper semilattice category (with zero morphisms) again. $\square$

¹ For set comprehension (and quantification) we shall use the notation of Z [17], which uses the pattern

$$\{ \text{declaration} \mid \text{predicate} \bullet \text{term} \} ,$$

to denote the set of all values of *term* under bindings for the locally bound variables from *declaration* that satisfy the *predicate* (which defaults to true), for example,

$$\{k : \mathbb{N} \mid k < 4 \bullet k^2\} = \{0, 1, 4, 9\}.$$

4 Signatures and Terms

For the sake of minimising notational overhead for the motivating example, we only consider single-sorted signatures. Also, since we do not need to distinguish constant symbols from zero-ary function symbols, we allow arbitrary natural numbers as arity of function symbols and do not consider separate constant symbols.

Definition 4.1. A *signature* $\Sigma = (\mathcal{F}, \text{arity})$ consists of a set $\mathcal{F}$ of *function symbols* and a total mapping $\text{arity} : \mathcal{F} \rightarrow \mathbb{N}$ assigning each function symbol the number of arguments it requires in term construction.

A signature is called *unary* if it contains only unary function symbols. $\square$

Given a signature Σ and a set (normally of variables) $\mathcal{X}$, we write $\mathcal{T}_\Sigma \mathcal{X}$ for the set of Σ -terms over $\mathcal{X}$. The *variable injection* $\mathbb{V}_{\Sigma, \mathcal{X}} : \mathcal{X} \hookrightarrow \mathcal{T}_\Sigma \mathcal{X}$ maps each variable x to the term x , and the inductively defined *free variable relation*

$$\begin{aligned} \mathbb{F}_{\Sigma, \mathcal{X}} & : \mathcal{T}_\Sigma \mathcal{X} \leftrightarrow \mathcal{X} \\ (x \mapsto y) \in \mathbb{F}_{\Sigma, \mathcal{X}} & \Leftrightarrow x = y \\ (f(t_1, \dots, t_n) \mapsto y) \in \mathbb{F}_{\Sigma, \mathcal{X}} & \Leftrightarrow (t_1 \mapsto y) \in \mathbb{F}_{\Sigma, \mathcal{X}} \vee \dots \vee (t_n \mapsto y) \in \mathbb{F}_{\Sigma, \mathcal{X}} \end{aligned}$$

relates each variable term with its free variables. (We occasionally omit subscripts where they can be inferred from the context.)

Given a relation $R : \mathcal{X} \leftrightarrow \mathcal{Y}$, we extend the term set construction to a relator by defining the morphism part inductively as the least relation $\mathcal{T}_\Sigma R$ satisfying:

$$\begin{aligned} (x \mapsto y) \in R & \Rightarrow (x \mapsto y) \in \mathcal{T}_\Sigma R \\ \{t_1 \mapsto u_1, \dots, t_n \mapsto u_n\} \subseteq R & \Rightarrow (f(t_1, \dots, t_n) \mapsto f(u_1, \dots, u_n)) \in \mathcal{T}_\Sigma R \end{aligned}$$

This is obviously monotonic, and also preserves identities and distributes over composition, so $\mathcal{T}_\Sigma$ is a relator.

Finally we need the “free extension” $\mathbb{E}_{\Sigma, \mathcal{X}} : \mathcal{T}_\Sigma(\mathcal{T}_\Sigma \mathcal{X}) \rightarrow \mathcal{T}_\Sigma \mathcal{X}$ which maps “terms over terms” to “terms over variables” by “flattening the structure”. It is easily verified that $\mathbb{V}_\Sigma$ and $\mathbb{E}_\Sigma$ are natural simulations; the remaining monad laws are equivalent to those for the standard categorical case:

Proposition 4.2. $(\mathcal{T}_\Sigma, \mathbb{V}_\Sigma, \mathbb{E}_\Sigma)$ is an ordered monad. $\square$

5 Relational Substitutions

Definition 5.1. Given two variable sets $\mathcal{X}$ and $\mathcal{Y}$, a *relational Σ -substitution* from $\mathcal{X}$ to $\mathcal{Y}$, written $\sigma : \mathcal{X} \rightharpoonup \mathcal{Y}$, is a relation $\sigma : \mathcal{X} \leftrightarrow \mathcal{T}_\Sigma \mathcal{Y}$.

The set of all relational Σ -substitutions from $\mathcal{X}$ to $\mathcal{Y}$ is written $\mathcal{X} \rightharpoonup \mathcal{Y}$. $\square$

The inclusion ordering $\subseteq$ on $\mathcal{X} \rightharpoonup \mathcal{Y}$, and therefore also meets and joins, are those of relations in $\mathcal{X} \leftrightarrow \mathcal{T}_\Sigma \mathcal{Y}$.

Since we have shown that the term functor $\mathcal{T}_\Sigma$ extends to an ordered monad, a relational substitution is a morphism of the Kleisli category $\mathbb{K} \mathcal{T}_\Sigma$, and Lemma 3.2 implies:

Proposition 5.2. Taking variable sets as objects and relational Σ -substitutions between them as morphisms produces an ordered category with domain, which we denote RelSubst_Σ , and which is defined as the Kleisli category of the ordered term monad. $\square$

Because of Lemma 3.2, domain minimality characterises exactly the univalent relational substitutions, and we use this to define the subcategory Subst_Σ which is equivalent to the (co-cartesian) category of standard substitutions with standard composition of substitutions:

Definition 5.3. Subst_Σ is the restriction of RelSubst_Σ to deterministic and total relational Σ -substitutions.

TotSubst_Σ is the restriction to total relational Σ -substitutions. $\square$

Since inclusion and meets are inherited from the underlying relations, and since meets are not subject to additional requirements in lower semilattice categories, RelSubst_Σ is even a lower semilattice category. It also has range, which identifies the free variables of the substitution:

Proposition 5.4. The ordered category RelSubst_Σ has range; for $\sigma : \mathcal{X} \rightarrow \mathcal{Y}$, we have: $\text{ran}_{\text{RelSubst}_\Sigma} \sigma = \text{ran}_{\text{Rel}} (\sigma : \mathbb{F}_{\Sigma, \mathcal{Y}}) : \mathbb{V}_{\Sigma, \mathcal{Y}}$ $\square$

Both Rel and $\mathcal{T}_\Sigma \text{Rel}$ have empty relations as least elements of their homsets, but if Σ has a zero-ary function symbol, say c , then the relator $\mathcal{T}_\Sigma$ does not preserve the least element, since $(c \mapsto c) \in \mathcal{T}_\Sigma \emptyset$. In such cases, empty morphisms in RelSubst_Σ are not zero morphisms — for example, we have

$$\{x \mapsto c()\} \circ \sigma = \{x \mapsto c()\}$$

for all relational substitutions σ , even when σ is empty.

However, if there are no zero-ary function symbols, then each term contains at least one variable, and $\mathcal{T}_\Sigma \emptyset = \emptyset$, so Lemma 3.4 implies:

Proposition 5.5. If Σ has no zero-ary function symbols, then the empty relational substitutions $\emptyset_{\mathcal{X}, \mathcal{Y}} : \mathcal{X} \rightarrow \mathcal{Y}$ are zero morphisms. $\square$

If f is a binary function symbol in Σ , and we consider the two relations $R = \{x \mapsto y\}$ and $S = \{x \mapsto z\}$, then the term $f(x, x)$ is associated

- by $\mathcal{T}_\Sigma R$ only with the term $f(y, y)$,
- by $\mathcal{T}_\Sigma S$ only with the term $f(z, z)$,
- by $\mathcal{T}_\Sigma (R \cup S)$ with the terms $f(y, y)$, $f(y, z)$, $f(z, y)$, and $f(z, z)$,

so in such cases, the term relator $\mathcal{T}_\Sigma$ does not even preserve binary joins.

However, if all function symbols are at most unary, then each term contains at most one variable, and the term relator $\mathcal{T}_\Sigma$ preserves all non-empty joins, in particular, $\mathcal{T}_\Sigma (R \cup S) = \mathcal{T}_\Sigma R \cup \mathcal{T}_\Sigma S$, and we easily obtain:

Proposition 5.6. If Σ has no function symbols with arity greater than 1, then composition distributes over non-empty joins to its right. $\square$

In the narrow space between these two classes of counterexample, we essentially obtain path languages, and Lemma 3.4 implies:

Proposition 5.7. If Σ contains only unary function symbols, then the category RelSubst_Σ is a complete Kleene category with domain and range. $\square$

For relations, R is deterministic iff R is univalent, and if $R : S$ is deterministic, then $\text{ran } R : S$ is deterministic, too, since $S^\sim : \text{ran } R : S \subseteq S^\sim : R^\sim : R : S \subseteq \mathbb{I}$. It is an interesting question for which more general structures the corresponding property holds; for relational substitutions it can be shown directly:

Lemma 5.8. If $\sigma \circ \tau$ is deterministic in RelSubst_Σ , then so is $\text{ran } \sigma \circ \tau$.

PROOF: If $\text{ran } \sigma \circ \tau$ is not deterministic, then there are a variable x and terms $t_1 \neq t_2$ such that $\{x \mapsto t_1, x \mapsto t_2\} \subseteq \text{ran } \sigma \circ \tau$.

Since $\mathbb{F}_\Sigma$ is the identity in the Kleisli category, we have, with Prop. 5.4,

$$\text{ran } \sigma \circ \tau = \text{ran } (\sigma : \mathbb{F}_\Sigma) : \mathbb{V} \circ \tau = \text{ran } (\sigma : \mathbb{F}_\Sigma) : \tau,$$

so there would then be a variable y and a term t such that $(y \mapsto t) \in \sigma$ and $(t \mapsto x) \in \mathbb{F}_\Sigma$. This implies $t[x \setminus t_1] \neq t[x \setminus t_2]$, which, because of $\{y \mapsto t[x \setminus t_1], y \mapsto t[x \setminus t_2]\} \subseteq \sigma \circ \tau$, shows that $\sigma \circ \tau$ is not univalent, either. $\square$

If we restrict attention to total relation substitutions, then meets do not generally exist, and the identity $\mathbb{V}_{\Sigma, \mathcal{X}}$ is the only subidentity on $\mathcal{X}$. Technically, this implies that TotSubst_Σ has domain and range, but for all morphisms, domain and range are identities.

Proposition 5.9. The category TotSubst_Σ with variable sets as objects and *total* relational substitutions between them as morphisms is an ordered category with domain and range, where, for $\sigma : \mathcal{X} \rightrightarrows \mathcal{Y}$, we have:

$$\text{dom } \sigma = \mathbb{V}_{\Sigma, \mathcal{X}} \qquad \text{ran } \sigma = \mathbb{V}_{\Sigma, \mathcal{Y}} \qquad \square$$

Perhaps surprisingly, this trivial domain operation still gives rise to a useful determinacy concept:

Lemma 5.10. A morphism $\sigma : \mathcal{X} \rightrightarrows \mathcal{Y}$ is domain-minimal in TotSubst_Σ iff it is univalent as a relation in $\mathcal{X} \leftrightarrow \mathcal{I}_\Sigma \mathcal{Y}$.

PROOF: From the definition of domain-minimality and the above definition of dom in TotSubst_Σ we obtain:

$$\begin{aligned} & \sigma : \mathcal{X} \rightrightarrows \mathcal{Y} \text{ is domain-minimal in } \text{TotSubst}_\Sigma \\ \Leftrightarrow & \quad \forall \rho : \mathcal{X} \rightrightarrows \mathcal{Y} \bullet \rho \subseteq \sigma \Rightarrow \rho = \text{dom } \rho \circ \sigma \\ \Leftrightarrow & \quad \forall \rho : \mathcal{X} \rightrightarrows \mathcal{Y} \bullet \rho \subseteq \sigma \Rightarrow \rho = \mathbb{V}_{\Sigma, \mathcal{X}} \circ \sigma \\ \Leftrightarrow & \quad \forall \rho : \mathcal{X} \rightrightarrows \mathcal{Y} \bullet \rho \subseteq \sigma \Rightarrow \rho = \sigma \end{aligned}$$

So σ is domain-minimal if it is $\subseteq$ -minimal in TotSubst_Σ , which holds exactly for mappings, i.e., for the univalent total relations. $\square$

The property of Lemma 5.8 however, does not carry over to TotSubst_Σ ; that would require further restriction to relational substitutions that are surjective in RelSubst_Σ , that is, with $\text{ran } \sigma = \mathbb{V}$.

6 Substitution Sets

From a category $\mathbf{C}$ for which the collection of morphisms between any two objects is a set (and not a class), one can construct a new category where the morphisms are subsets of the homsets of $\mathbf{C}$:

Definition 6.1. For a locally small category $\mathbf{C}$, we define the *morphism set category* $\mathbb{P}\mathbf{C}$ as follows:

- The objects of $\mathbb{P}\mathbf{C}$ are the same as the objects of $\mathbf{C}$.
- A morphism in $\mathbb{P}\mathbf{C}$ from $\mathcal{A}$ to $\mathcal{B}$ is a set of morphisms in $\mathbf{C}$ from $\mathcal{A}$ to $\mathcal{B}$.
- For an object $\mathcal{A}$, the singleton set $\{\mathbb{I}_{\mathcal{A}}\}$ is the identity on $\mathcal{A}$ in $\mathbb{P}\mathbf{C}$.
- $\mathcal{R} ; \mathcal{S} := \{R \in \mathcal{R}; S \in \mathcal{S} \bullet R; S\}$ $\square$

The following seems to be a folklore theorem; it is also not hard to show:

Fact 6.2. For any locally small category $\mathbf{C}$, the morphism set category $\mathbb{P}\mathbf{C}$ is a complete Kleene category with respect to the subset ordering. $\square$

For a free monoid represented as a one-object category $\mathcal{M}$, and generated from the alphabet $\mathcal{A}$, the morphism set category $\mathbb{P}\mathcal{M}$ is the Kleene algebra of regular languages over $\mathcal{A}$.

Note that such morphism set categories have only two subidentities on each object, namely the identity $\{\mathbb{I}_{\mathcal{A}}\}$ and the zero endomorphism $\{\}$. Therefore, morphism set categories have domain and range, but they are of somewhat limited usefulness.

However, we have $\text{dom } R = \{\} \Leftrightarrow R = \{\}$, and therefore domain minimality does characterise exactly the set morphisms with at most one element:

$$\begin{aligned} & \forall R : \mathcal{A} \rightarrow \mathcal{B} \bullet R \subseteq F \Rightarrow R = \text{dom } R ; F \\ \Leftrightarrow & \quad \forall R : \mathcal{A} \rightarrow \mathcal{B} \bullet R \subseteq F \Rightarrow (R = \{\} \vee R = F) \end{aligned}$$

We use the morphism set construction on total and univalent relational substitutions:

Definition 6.3. We define the Σ -substitution set category over a signature Σ as $\text{SubstSet}_\Sigma := \mathbb{P}\text{Subst}_\Sigma$.

Using set union $\bigcup$ on substitution sets considered as sets of univalent relations maps each morphism $\mathcal{R} : \mathcal{A} \rightarrow \mathcal{B}$ in SubstSet_Σ to a relational substitution $(\bigcup \mathcal{R}) : \mathcal{A} \rightarrow \mathcal{B}$.

This is not a functor, since in general, we only have $\bigcup(\mathcal{R}; \mathcal{S}) \subseteq \bigcup \mathcal{R} \circ \bigcup \mathcal{S}$, but not equality, for example:

$$\begin{aligned}
 & \bigcup\{\{x \mapsto f(y, y)\}\} \circ \bigcup\{\{y \mapsto a\}, \{y \mapsto b\}\} \\
 = & \{x \mapsto f(y, y)\} \circ \{y \mapsto a, y \mapsto b\} \\
 = & \{x \mapsto f(a, a), x \mapsto f(a, b), x \mapsto f(b, a), x \mapsto f(b, b)\} \\
 \neq & \{x \mapsto f(a, a), x \mapsto f(b, b)\} \\
 = & \bigcup\{\{x \mapsto f(a, a)\}, \{x \mapsto f(b, b)\}\} \\
 = & \bigcup\{\{x \mapsto f(y, y)\} \circ \{y \mapsto a\}, \{x \mapsto f(y, y)\} \circ \{y \mapsto b\}\} \\
 = & \bigcup(\{\{x \mapsto f(y, y)\}\} ; \{\{y \mapsto a\}, \{y \mapsto b\}\})
 \end{aligned}$$

The natural mapping in the converse direction, namely, mapping each total relational substitution $R : \mathcal{A} \xrightarrow{\Sigma} \mathcal{B}$ to the non-empty set $\text{Maps}(R)$ of all total and deterministic substitutions contained in R , is not a functor either, as can be seen from the same example.

7 Unification Via Determinisation

A unification problem is normally represented as an (injective) sequence of equations in $\mathcal{T}_{\Sigma} \mathcal{X}$

$$U = \langle l_1 = r_1, \dots, l_n = r_n \rangle .$$

We will call this a “conventional unification problem”.

To be able to deal with this inside our substitution categories, we first define a variable set

$$\mathcal{E}_n := \{e_1, \dots, e_n\}$$

with pairwise distinct variables $e_1, \dots, e_n$ serving as identifiers for the equations.

Now we can create two univalent relational substitutions collecting all the left-hand sides, respectively all the right-hand sides (“ $\#U$ ” denotes the cardinality of the set U):

$$\begin{aligned}
 \lambda_U, \rho_U : \mathcal{E}_{\#U} & \xrightarrow{\Sigma} \mathcal{X} & \lambda_U &:= \{i : 1.. \#U \bullet e_i \mapsto l_i\} \\
 & & \rho_U &:= \{i : 1.. \#U \bullet e_i \mapsto r_i\}
 \end{aligned}$$

We can collect these into a two-element substitution set, or into a single relational substitution:

$$\begin{aligned}
 \mathcal{H}_U : \mathcal{E}_{\#U} & \rightarrow \mathcal{X} & \mathcal{H}_U &:= \{\lambda_U, \rho_U\} \\
 \eta_U : \mathcal{E}_{\#U} & \xrightarrow{\Sigma} \mathcal{X} & \eta_U &:= \lambda_U \cup \rho_U = \bigcup \mathcal{H}_U
 \end{aligned}$$

The standard definition of unification specifies the most general unifier μ_U for U as an co-equaliser for λ_U and ρ_U in the category $\mathbf{Subst}_{\Sigma}$, i.e., μ_U is a total and univalent substitution such that $\lambda_U \circ \mu_U = \rho_U \circ \mu_U$, and for any ν with $\lambda_U \circ \nu = \rho_U \circ \nu$ there exists a unique ϕ such that $\nu = \mu_U \circ \phi$.

For moving this into the relational setting, we will consider *deterministic*, i.e., domain-minimal, morphisms.

Definition 7.1. In an ordered category with domain, we call a morphism M a *determiniser* for another morphism R iff $R \circ M$ is deterministic. $\square$

In [3] it is shown that subidentities are deterministic, and also that morphisms contained in deterministic morphisms are deterministic as well. From monotonicity of composition we then immediately obtain:

Lemma 7.2. If M is a determiniser for R , and $M' \subseteq M$, then M' is a determiniser for R , too. $\square$

We now explore how the concept of “most general unifier” can be transferred into the relational setting. As a first attempt, we directly transfer the co-equaliser-based definition:

Definition 7.3. In an ordered category with domain, an *initial determiniser* for a morphism R is a determiniser M for R such that for every other determiniser M' for R , there is exactly one morphism Φ such that $M' = M \circ \Phi$. $\square$

This choice of terminology follows Goguen’s presentation of most-general unifiers [9], and is justified by considering the category where objects are determinisers for R , and morphisms from a determiniser $M : \mathcal{B} \rightarrow \mathcal{C}$ to another determiniser $M' : \mathcal{B} \rightarrow \mathcal{C}'$ are morphisms $F : M \rightarrow M'$ for which $M \circ F = M'$.

7.1 Initial Determinisers for Substitution Sets

Let us first investigate the situation in the substitution set category $\text{SubstSet}_{\mathcal{D}}$.

For a most general unifier μ_U for U , the singleton set $\{\mu_U\}$ is a deterministic substitution set, and the composition

$$\mathcal{H}_U \circ \{\mu_U\} = \{\lambda_U, \rho_U\} \circ \{\mu_U\} = \{\lambda_U \circ \mu_U, \rho_U \circ \mu_U\} = \{\lambda_U \circ \mu_U\}$$

is deterministic, too, so $\{\mu_U\}$ is a determiniser for $\mathcal{H}_U$.

Furthermore, for *any* determiniser $\mathcal{N}$ for $\mathcal{H}_U$, we have that for each $\nu \in \mathcal{N}$,

$$\lambda_U \circ \nu = \rho_U \circ \nu ,$$

so there is a ϕ_ν such that $\nu = \mu_U \circ \phi_\nu$. Then we have

$$\mathcal{N} = \{\nu : \mathcal{N} \bullet \nu\} = \{\nu : \mathcal{N} \bullet \mu_U \circ \phi_\nu\} = \{\mu_U\} \circ \{\nu : \mathcal{N} \bullet \phi_\nu\} ,$$

so, with $\Phi := \{\nu : \mathcal{N} \bullet \phi_\nu\}$, we have $\mathcal{N} = \{\mu_U\} \circ \Phi$.

Now assume for an appropriate substitution set Φ' that $\{\mu_U\} \circ \Phi' = \mathcal{N}$ holds; from the definition of composition “ $\circ$ ” we have:

$$\begin{aligned} \{\mu_U\} \circ \Phi' \supseteq \mathcal{N} &\Leftrightarrow \forall \nu : \mathcal{N} \bullet \exists \phi : \Phi' \bullet \mu_U \circ \phi = \nu \\ &\Rightarrow \forall \nu : \mathcal{N} \bullet \phi_\nu \in \Phi' \\ &\Leftrightarrow \Phi \subseteq \Phi' \end{aligned}$$

and

$$\begin{aligned}
 \{\mu_U\} ; \Phi' \subseteq \mathcal{N} &\Leftrightarrow \forall \phi : \Phi' \bullet \mu_U \circ \phi \in \mathcal{N} \\
 &\Leftrightarrow \forall \phi : \Phi' \bullet \exists \nu : \mathcal{N} \bullet \mu_U \circ \phi = \nu \\
 &\Rightarrow \forall \phi : \Phi' \bullet \exists \nu : \mathcal{N} \bullet \phi = \phi_\nu \\
 &\Leftrightarrow \Phi' \subseteq \Phi ,
 \end{aligned}$$

which implies that Φ is the uniquely determined substitution set such that $\mathcal{N} = \{\mu_U\} ; \Phi$.

With all this, we have shown the following:

Theorem 7.4. If μ is a most general unifier for U , then, in SubstSet_Σ , $\{\mu\}$ is an initial determiniser for $\mathcal{H}_U$. $\square$

Now assume the $\mathcal{M}$ is any substitution set which is a determiniser for $\mathcal{H}$ and uniquely factors each other determiniser $\mathcal{N}$ over $\Phi_{\mathcal{N}}$.

If $\mu, \mu_0 \in \mathcal{M}$, then, according to Lemma 7.2, $\{\mu\}$ is a determiniser for $\mathcal{H}$, too, and we have:

$$\{\mu\} = \mathcal{M} ; \Phi_{\{\mu\}} = \{\mu_0\} ; \Phi_{\{\mu\}}$$

Therefore, $\mathcal{M}$ factors over any of its atoms $\{\mu_0\}$:

$$\mathcal{M} = \bigcup \{\mu : \mathcal{M} \bullet \{\mu\}\} = \bigcup \{\mu : \mathcal{M} \bullet \{\mu_0\} ; \Phi_{\{\mu\}}\} = \{\mu_0\} ; \bigcup \{\mu : \mathcal{M} \bullet \Phi_{\{\mu\}}\}$$

For any Φ such that $\mathcal{M} = \{\mu_0\} ; \Phi$ we have

$$\mathcal{M} = \{\mu_0\} ; \Phi = \mathcal{M} ; \Phi_{\{\mu_0\}} ; \Phi$$

Since $\mathcal{M} ; \{\mathbb{V}\} = \mathcal{M} ; \mathbb{I} = \mathcal{M}$ and we have unique factorisation, we also have $\Phi_{\{\mu_0\}} ; \Phi = \{\mathbb{V}\}$. Therefore, all elements of $\Phi_{\{\mu_0\}}$ and of Φ must be variable permutations, and if we have $\alpha, \beta \in \Phi_{\{\mu_0\}}$ and $\gamma, \delta \in \Phi$, then

$$\alpha \circ \gamma = \alpha \circ \delta = \beta \circ \gamma = \beta \circ \delta = \mathbb{V}$$

Then, using inverses of variable permutations, we have $\alpha = \gamma^{-1} = \beta$ and $\gamma = \alpha^{-1} = \delta$, so $\Phi_{\{\mu_0\}}$ and Φ are both one-element sets, and Φ is uniquely determined as containing the inverse of the single element ϕ_{μ_0} of $\Phi_{\{\mu_0\}}$.

Using this in the introducing assumption for Φ , we obtain:

$$\mathcal{M} = \{\mu_0\} ; \Phi = \{\mu_0\} ; \{\phi_{\mu_0}^{-1}\} = \{\mu_0 \circ \phi_{\mu_0}^{-1}\} ,$$

and altogether we now have shown the following:

Theorem 7.5. In SubstSet_Σ , a non-empty initial determiniser for a substitution set $\mathcal{H}$ consists of a single element which is a most general unifier for all elements of $\mathcal{H}$. $\square$

An empty initial determiniser for $\mathcal{H}$ in SubstSet_Σ is obviously the only determiniser for $\mathcal{H}$ and indicates that the substitutions which are the elements of $\mathcal{H}$ do not have a unifier.

Since any substitution set is a determiniser for the empty substitution set $\emptyset$, any isomorphism starting from $\mathcal{Y}$ is an initial determiniser for $\emptyset : \mathcal{X} \rightarrow \mathcal{Y}$ in SubstSet_Σ .

Since even an infinite set of terms has a most general unifier if it unifiable, as shown by Marciniak [15], we altogether obtain:

Corollary 7.6. Every morphism $\mathcal{H}$ in SubstSet_Σ has an initial determiniser, which is either a singleton $\{\mu\}$, with μ being a most general unifier for all the elements of $\mathcal{H}$, or empty iff there is no such unifier. $\square$

7.2 Initial Total Determinisers for Relational Substitutions

Now we direct our attention to the situation in the category RelSubst_Σ of relational substitutions. Note that since the homsets of RelSubst_Σ are atomic Boolean lattices, the composition of deterministic relational substitutions is deterministic again — the argument of [3, Lemma 23] can be lifted to the setting here; alternatively, this could also be shown directly via univalence.

A most general unifier μ_U for a conventional unification problem U is, by definition, a deterministic relational substitution, and the composition

$$\eta_U \circ \mu_U = (\lambda_U \cup \rho_U) \circ \mu_U = \lambda_U \circ \mu_U \cup \rho_U \circ \mu_U = \lambda_U \circ \mu_U$$

is, as composition of deterministic relational substitutions, deterministic, too, so μ_U is a determiniser for η_U .

According to Lemma 7.2, each $\mu_0 \subseteq \mu_U$ is a determiniser for η_U , too, but we do not necessarily have factorisation for μ_0 — for example, for

$$\begin{aligned} \mu_U &= \{x \mapsto f(z), y \mapsto g(z)\} \\ \mu_0 &= \{x \mapsto f(z)\} \end{aligned}$$

there is no relational substitution ϕ such that $\mu_0 = \mu_U \circ \phi$.

Additionally, whenever $\text{dom } \mu$ is strictly smaller than $\text{dom } \mu_{U_1}$, there can be no relational substitution ϕ such that $\mu_1 = \mu \circ \phi$.

Therefore, it makes sense to restrict attention to total determinisers here:

Definition 7.7. A *initial total determiniser* for a morphism R is a *total* determiniser M for R such that for every other *total* determiniser M' for R , there is exactly one *total* morphism Φ such that $M' = M \circ \Phi$. $\square$

Now consider any determiniser ν for η_U . Since $\eta_U \circ \nu$ is deterministic, $\text{ran } \eta_U \circ \nu$ is deterministic, too, according to Lemma 5.8, and it obviously is also a unifier for λ_U and ρ_U .

However, $\text{ran } \eta_U \circ \nu$ is total if and only if η_U is surjective, i.e., if $\text{ran } \eta_U = \mathbb{V}$. If η_U is not surjective, then ν is not necessarily deterministic — examples are easy to construct.

Let us therefore first consider the case where η_U is surjective, and therefore ν is deterministic. Then the fact that μ_U is a most general unifier provides a

unique total deterministic relational substitution ϕ such that $\nu = \mu_U \circ \phi$. Since, with a standard argument, μ_U is surjective, any ψ with $\nu = \mu_U \circ \psi$ has to be deterministic, so ϕ is also the unique total (not necessarily deterministic) relational substitution factoring ν .

If $\eta_U : \mathcal{X} \rightarrow \mathcal{Y}$ is not surjective, then it is possible to represent $\mathcal{Y}$ as a co-product² $\mathcal{Y}_1 \xrightarrow{\iota_1} \mathcal{Y} \xleftarrow{\iota_2} \mathcal{Y}_2$ with injections ι_1 and ι_2 , such that $\text{ran } \eta_U = \text{ran } \iota_1$. Then, using a standard argument to show in particular disjointness of the ranges of the two components, $\mu_U = \mu_1 + \mu_2$ where μ_1 is a most general unifier for $\eta_U \circ \iota_1$, which is surjective, and μ_2 is an isomorphism (i.e., a bijective variable renaming). Then we obtain a unique factoring $\iota_1 \circ \nu = \mu_1 \circ \phi$, and another unique factoring $\iota_2 \circ \nu = \mu_2 \circ \mu_2' \circ \iota_2 \circ \nu$, so that altogether we have the unique factoring $\nu = [\phi, \mu_2' \circ \iota_2 \circ \nu]$. This shows:

Theorem 7.8. If μ is a most general unifier for U , then, in RelSubst_Σ , μ is an initial total determiniser for η_U . $\square$

Now, for a total and surjective relational substitution η , assume that μ is a total determiniser such that for each other total determiniser ν there is a unique total relational substitution ϕ_ν such that $\nu = \mu \circ \phi_\nu$.

Since η is surjective, Lemma 5.8 implies that μ and ν are both deterministic. Since, by the standard argument, μ is surjective, too, also ϕ_ν has to be deterministic. For any total and deterministic relational substitution $\eta_0 \subseteq \eta$ we then have $\eta_0 \circ \mu \subseteq \eta \circ \mu$, and since both sides are total and deterministic, we have equality. Therefore, μ is a most general unifier for all total and deterministic relational substitutions contained in η .

On the other hand, if $\eta : \mathcal{E} \rightarrow \mathcal{Y}$ is an empty relation, then it is easy to see that exactly isomorphisms starting at $\mathcal{Y}$ are initial total determinisers.

Any non-surjective $\eta : \mathcal{E} \rightarrow \mathcal{Y}$ can again be split via a direct sum into a surjective part $\eta \circ \iota_1$ and an empty part $\eta \circ \iota_2$, and we obtain:

Theorem 7.9. In RelSubst_Σ , every initial total determiniser for a total relational substitution η is a deterministic total relational substitution which is a most general unifier for all deterministic total relational substitutions contained in η . $\square$

From Theorem 7.4 and Theorem 7.5 we see that the statements of Theorem 7.8 and Theorem 7.9 also hold in the substitution set setting SubstSet_Σ . This demonstrates that Def. 7.7 of *initial total determiniser* is a plausible abstraction of the concept of “most general unifier”, and relates usefully with that concept in the quite different settings of RelSubst_Σ and SubstSet_Σ .

This determiniser concept even has a useful meaning for more “standard” relations; we give a name to the required setting:

Definition 7.10. A *Kleene allegory* is a distributive allegory which is also a Kleene category. $\square$

² In RelSubst_Σ , which coincides with a direct sum in Rel .

(Allegories provide meet, converse, domain, and range, and determinism is equivalent to univalence; distributive allegories add zero morphisms, join and distributivity laws.) Adding residuals and completeness to Kleene allegories would produce complete Dedekind categories, or “heterogeneous relation algebras without complement”.

Theorem 7.11. In a Kleene allegory with quotients, an initial total determiniser of a morphism $R : \mathcal{A} \rightarrow \mathcal{B}$ is a quotient projection for the equivalence relation $(R^\sim; R)^*$.

PROOF: The morphism $\chi : \mathcal{B} \rightarrow \mathcal{Q}$ is a quotient projection for an equivalence relation Ξ iff $\chi; \chi^\sim = \Xi$ and $\chi^\sim; \chi = \mathbb{I}_{\mathcal{Q}}$, so χ is total and deterministic by definition.

If χ is a quotient projection for $\Xi := (R^\sim; R)^*$, then $R; \chi$ is univalent:

$$\chi^\sim; R^\sim; R; \chi \subseteq \chi^\sim; \Xi; \chi = \chi^\sim; \chi; \chi^\sim; \chi = \mathbb{I}_{\mathcal{Q}},$$

so χ is a determiniser for R .

If $\mu : \mathcal{B} \rightarrow \mathcal{C}$ is any total determiniser for R , then we have $\mu = \Xi; \mu$:

$$\begin{array}{ll} \mu \subseteq \Xi; \mu & \Xi \text{ is reflexive} \\ = (R^\sim; R)^*; \mu & \text{Def. } \Xi \\ \subseteq (\mu; \mu^\sim; R^\sim; R)^*; \mu & \mu \text{ is total} \\ = \mu; (\mu^\sim; R^\sim; R; \mu)^* & \text{properties of refl. trans. closure} \\ \subseteq \mu; \mathbb{I}_{\mathcal{C}}^* & \mu \text{ is determiniser for } R \\ = \mu & \end{array}$$

Therefore, μ factors over χ as $\mu = \Xi; \mu = \chi; \chi^\sim; \mu$:

If $\mu = \chi; \phi$ is any factoring over χ , then $\phi = \chi^\sim; \chi; \phi = \chi^\sim; \mu$, so we have unique factoring, and χ is an initial total determiniser.

If μ is an initial total determiniser, then μ is isomorphic to χ and therefore a quotient projection, too. $\square$

8 Related Work

Rydeheard and Burstall [16] and Goguen [9] (who used the dual setting) pointed out that unification corresponds to determining co-equalisers in the Kleisli category of the term monad.

Instead of using an ordered monad, relational substitutions can also be obtained as morphisms in the Kleisli category of the composition of the powerset monad with the term monad. Monad composition does work under certain conditions, several of these were developed by Jones and Duponcheel [10], one of them being the presence of a “distributive law” originally proposed by Beck [2], or equivalently a “swapper” natural transformation, which Eklund *et al.* [5] use to show that the composition $\mathcal{T}_{\Sigma}; \mathbb{P}$ of the term functor with the powerset functor can be extended to a monad, too. Note that arbitrary monads cannot necessarily

be composed to a new monad as shown by Jones and Duponcheel [10]. The string rewriting approach of that proof is explicitly elaborated by Kozen [13] to produce a general tool for verifying monad compositions and re-prove the monadicity of $\mathcal{T}_\Sigma : \mathbb{P}$. Eklund *et al.* replaced the standard powerset monad $\mathbb{P}$ with L -fuzzy powerset monads in [7].

Eklund and Gähler use a “partially ordered monad” concept restricted to endofunctors on **Set** and show that under certain conditions the resulting Kleisli category is a Kleene category [8]. These conditions make intrinsic use of **Set** structure and establish the result by guaranteeing that the Kleisli category is a complete upper semilattice category.

Where Eklund *et al.* proceed to use the composed monad for unification [6], they consider equations consisting of *two* relational substitutions, just like previous work on unification in the categorical context.

9 Conclusion

For a relatively general kind of relational categories, we introduced the concept of *determiniser* which enables treatment of unification problems represented as a *single* relational morphism.

By discussing RelSubst_Σ and SubstSet_Σ in some detail, we gave a first flavour of the effects involved in relational generalisations of substitutions. The discussion of TotSubst_Σ showed that even such a seemingly simple variation can produce a quite different setting.

The discussion of SubstSet_Σ was greatly simplified by the fact that it could be defined as the morphism set category of the *trivially ordered* category Subst_Σ ; “multi-substitutions” could be defined by using, for example, TotSubst_Σ as basis for the morphism set category construction, which would then need to be equipped with a more complex ordering in its homsets.

Similarly, using categories of L -fuzzy relations [18] instead of **Rel** as basis for RelSubst_Σ would open up exploration of these issues in a “generalised relations” setting closely related to that of Eklund *et al.* [7].

However, we feel that the development here shows that ordered monads are an attractive alternative to monad composition.

The use of domain minimality as determinacy concept seems to be quite a natural fit, and is an important ingredient for turning the apparently weak theory of ordered categories with domain into a powerful abstraction tool.

Further algebraic exploration of determinisation will also require further exploration of properties of domain minimality; other explorations in ordered categories or Kleene algebras with domain may well find that substitutions provide interesting example models.

I am grateful to the anonymous reviewers for their constructive and useful comments.

³ For the term monad as presented there it is not clear how the subterm ordering gives rise even to an almost-complete semilattice (Example 2), nor what the least element “ $\emptyset$ ” required for Example 6 (Kleene monad) might be.

References

1. Backhouse, R.C.: Constructive Lattice Theory (1993), <http://www.cs.nott.ac.uk/~rcb/papers/abstract.html#isos>
2. Beck, J.: Distributive laws. In: Appelgate, H., Eckmann, B. (eds.) Seminar on triples and categorical homology theory, ETH, 1966-67. Lect. Notes in Math., vol. 80, pp. 119–140. Springer, Heidelberg (1969)
3. Desharnais, J., Möller, B.: Characterizing Determinacy in Kleene Algebras. *Information Sciences* 139, 253–273 (2001)
4. Desharnais, J., Möller, B., Struth, G.: Kleene Algebra with Domain. *ACM Transactions on Computational Logic* 7(4), 798–833 (2006)
5. Eklund, P., Galán, M.A., Ojeda-Aciego, M., Valverde, A.: Set functors and generalised terms. In: IPMU 2000, 8th Information Proc. and Management of Uncertainty in Knowledge-Based Systems Conference, vol. III, pp. 1595–1599 (2000)
6. Eklund, P., Galán, M.A., Medina, J., Ojeda Aciego, M., Valverde, A.: A categorical approach to unification of generalised terms. *Electronic Notes in Computer Science*, 66(5), 41–51 (2002), Special Issue: UNCL 2002, Unification in Non-Classical Logics (ICALP 2002 Satellite Workshop)
7. Eklund, P., Galán, M.A., Medina, J., Ojeda Aciego, M., Valverde, A.: Set functors, L -Fuzzy Set Categories, and Generalized Terms. *Computers and Mathematics with Applications* 43(6–7), 693–705 (2002)
8. Eklund, P., Gähler, W.: Partially ordered monads and powerset Kleene algebras. In: Proc. 10th Information Processing and Management of Uncertainty in Knowledge Based Systems Conference (IPMU 2004) (2004)
9. Goguen, J.A.: What is Unification? In: Ait-Kaci, H., Nivat, M. (eds.) *Resolution of Equations in Algebraic Structures, Algebraic Techniques*, vol. 1, pp. 217–261. Academic Press, Boston (1989)
10. Jones, M.P., Duponcheel, L.: Composing Monads. Research Report YALEU/DCS/RR-1004, Yale University, New Haven, Connecticut, USA (1993)
11. Kahl, W.: Refactoring Heterogeneous Relation Algebras around Ordered Categories and Converse. *J. Relational Methods in Comp. Sci.* 1, 277–313 (2004)
12. Kawahara, Y.: Notes on the Universality of Relational Functors. *Mem. Fac. Sci. Kyushu Univ. Ser. A* 27(2), 275–289 (1973)
13. Kozen, D.: Natural Transformations as Rewrite Rules and Monad Composition. Techn. Rep. TR2004-1942, Computer Science Dept., Cornell University (2004)
14. Lawvere, F.W.: Functorial Semantics of Algebraic Theories. *Proc. Nat. Acad. Sci. USA* 50, 869–872 (1963)
15. Marciniak, J.: Infinite Set Unification with Application to Categorical Grammar. *Studia Logica* 58, 339–355 (1997)
16. Rydeheard, D., Burstall, R.: A categorical unification algorithm. In: Poigné, A., Pitt, D.H., Rydeheard, D.E., Abramsky, S. (eds.) *Category Theory and Computer Programming. LNCS*, vol. 240, pp. 493–505. Springer, Heidelberg (1986)
17. Spivey, J.M.: *The Z Notation: A Reference Manual*, Prentice Hall International Series in Computer Science, 2nd edn. Prentice Hall, Englewood Cliffs (1992), <http://spivey.oriel.ox.ac.uk/~mike/zrm/>
18. Winter, M.: Goguen Categories: A Categorical Approach to L -fuzzy Relations. In: *Trends in Logic*, vol. 25 (2007)

Boolean Algebras and Stone Maps in Schröder Categories

Yasuo Kawahara

Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
kawahara@i.kyushu-u.ac.jp

Abstract. This paper concerns the concepts of Boolean algebras, the filters and Stone maps in Schröder categories, and further the development of the relational methodology, which might be the foundations of mathematics and computer science.

1 Introduction

Boolean algebras represent one of the most important concepts in mathematics and theoretical computer science. For example, relation algebras [10,13] are viewed as Boolean algebras with operators, and contact algebras [2] are Boolean algebras with relations satisfying suitable conditions. So far, the representation theorems of relation algebras are mainly based on atoms and Stone maps in Boolean algebras. For the sake of much more formality of these concepts, it is very interesting to re-formulate the concepts of Boolean algebras and the filters, and to demonstrate the representation theorems in Dedekind and Schröder categories [3,9]. Along the ideas above, the author tried to re-formulate the concepts of lattices and groups in relational categories [5,6]. In this paper the author aims to study a relational theory on Boolean algebras and the filters, and to show the representation theorems by atoms and Stone maps.

2 Dedekind and Schröder Categories

In this section we recall Dedekind categories [5,9] and Schröder categories [7,9,10], namely two kinds of relation categories.

Throughout this paper, a morphism α from an object X into an object Y in a Dedekind or Schröder category (defined below) will be denoted by a half arrow $\alpha : X \rightarrow Y$, and the composition of a morphism $\alpha : X \rightarrow Y$ followed by a morphism $\beta : Y \rightarrow Z$ will be written as $\alpha\beta : X \rightarrow Z$. Also we will denote the identity morphism on X as id_X .

Definition 1. A Dedekind category $\mathcal{D}$ is a category satisfying the following four conditions:

DC1. [Complete Heyting Algebra] For all pairs of objects X and Y the hom-set $\mathcal{D}(X, Y)$ consisting of all morphisms of X into Y is a complete Heyting algebra

with the least morphism $\emptyset_{XY}$ and the greatest morphism ∇_{XY} . Its algebraic structure will be denoted by

$$\mathcal{D}(X, Y) = (\mathcal{D}(X, Y), \sqsubseteq, \sqcup, \sqcap, \Rightarrow, \emptyset_{XY}, \nabla_{XY}),$$

where $\sqsubseteq, \sqcup, \sqcap$ and $\Rightarrow$ denote the inclusion order, the join, the meet and the relatively pseudo-complement of morphisms, respectively.

DC2. [Converse] There is given a converse operation $\sharp : \mathcal{D}(X, Y) \rightarrow \mathcal{D}(Y, X)$. That is, for all morphisms $\alpha, \alpha' : X \rightarrow Y$ and $\beta : Y \rightarrow Z$, the converse laws hold: (a) $(\alpha\beta)^\sharp = \beta^\sharp\alpha^\sharp$, (b) $(\alpha^\sharp)^\sharp = \alpha$, (c) If $\alpha \sqsubseteq \alpha'$, then $\alpha^\sharp \sqsubseteq \alpha'^\sharp$.

DC3. [Dedekind Formula] For all morphisms $\alpha : X \rightarrow Y$, $\beta : Y \rightarrow Z$ and $\gamma : X \rightarrow Z$ the Dedekind formula $\alpha\beta \sqcap \gamma \sqsubseteq \alpha(\beta \sqcap \alpha^\sharp\gamma)$ holds.

DC4. [Residual Composition] For all morphisms $\alpha : X \rightarrow Y$ and $\beta : Y \rightarrow Z$ the residual composition $\alpha \odot \beta : X \rightarrow Z$ is a morphism such that $\gamma \sqsubseteq \alpha \odot \beta$ if and only if $\alpha^\sharp\gamma \sqsubseteq \beta$ for all morphisms $\gamma : X \rightarrow Z$. $\square$

A Dedekind category is an abstraction of the categories of all binary relations and all fuzzy relations among sets. In what follows, the word *relation* is a synonym for morphism of Dedekind categories.

In a Dedekind category $\mathcal{D}$ the converse operation $\sharp : \mathcal{D}(X, Y) \rightarrow \mathcal{D}(Y, X)$ is an involutive bijection preserving $\sqsubseteq$ and so it holds that $\emptyset_{XY}^\sharp = \emptyset_{YX}$, $\nabla_{XY}^\sharp = \nabla_{YX}$, $(\sqcup_j \alpha_j)^\sharp = \sqcup_j \alpha_j^\sharp$ and $(\sqcap_j \alpha_j)^\sharp = \sqcap_j \alpha_j^\sharp$. Consequently DC3 $^\sharp$. $\alpha\beta \sqcap \gamma \sqsubseteq (\alpha \sqcap \gamma\beta^\sharp)\beta$ is valid. An object I of a Dedekind category $\mathcal{D}$ is called a (strict) *unit* if $\emptyset_{II} \neq \text{id}_I = \nabla_{II}$ and $\nabla_{XI}\nabla_{IX} = \nabla_{XX}$ for all objects X . A relation $f : X \rightarrow Y$ is called a *function*, denoted by $f : X \rightarrow Y$, if it is *univalent* ($f^\sharp f \sqsubseteq \text{id}_Y$) and *total* ($\text{id}_X \sqsubseteq ff^\sharp$). The universal relation $\nabla_{XI} : X \rightarrow I$ and the identity relation $\text{id}_X : X \rightarrow X$ are functions. A function $f : X \rightarrow Y$ is called an *injection* if $ff^\sharp = \text{id}_X$. Also a function $f : X \rightarrow Y$ is called a *surjection* if $f^\sharp f = \text{id}_Y$. An *I-point* x of X is a function $x : I \rightarrow X$. For a relation $\rho : I \rightarrow X$ the notation $x \in \rho$ will denote that x is an *I-point* with $x \sqsubseteq \rho$. The *domain* $\text{dom}(\alpha)$ of a relation $\alpha : X \rightarrow Y$ is a relation defined by $\text{dom}(\alpha) = \alpha\alpha^\sharp \sqcap \text{id}_X$.

The residual composition will be frequently used in the paper. For example, the supremum relation $\sup(\rho, \xi) : V \rightarrow X$ is defined by

$$\sup(\rho, \xi) = (\rho \odot \xi) \sqcap [(\rho \odot \xi) \odot \xi^\sharp]$$

for a pair of relations $\rho : V \rightarrow X$ and $\xi : X \rightarrow X$.

A Schröder category are a particular Dedekind category whose hom-sets are complete Boolean algebras.

Definition 2. A Schröder category $\mathcal{S}$ is a category satisfying the following two conditions SC1 and SC4 in addition to DC2 and DC3 in Def. $\square$

SC1. [Complete Boolean Algebra] For all pairs of objects X and Y the hom-set $\mathcal{S}(X, Y)$ consisting of all relations of X into Y is a complete Boolean algebra with the least relation $\emptyset_{XY}$ and the greatest relation ∇_{XY} . Its algebraic structure will be denoted by

$$\mathcal{S}(X, Y) = (\mathcal{S}(X, Y), \sqsubseteq, \sqcup, \sqcap, ^-, \emptyset_{XY}, \nabla_{XY}),$$

where $\sqsubseteq, \sqcup, \sqcap$ and $-$ denote the inclusion order, the join, the meet and the complement of relations, respectively.

SC4. [Zero Relation] The least relation $\emptyset_{XY}$ is a zero relation, that is, $\alpha \emptyset_{YZ} = \emptyset_{XZ}$. $\square$

The basic properties of Dedekind and Schröder categories are listed in Lemma 2.2 and Proposition 2.3 in [7]. To make our discussion richer we further impose the following four additional conditions on relational categories.

(RAT) [Rationality] For all relations $\alpha : X \rightarrow Y$ there exists a pair of functions $f : R \rightarrow X$ and $g : R \rightarrow Y$ such that $\alpha = f^\sharp g$ and $ff^\sharp \sqcap gg^\sharp = \text{id}_R$.

(PW) [Power Objects] For all objects Y there exists an object $\wp(Y)$ together with a membership relation $\ni_Y : \wp(Y) \rightarrow Y$ such that for all relations $\alpha : X \rightarrow Y$ there is a unique function $\alpha^\oplus : X \rightarrow \wp(Y)$ such that $\alpha = \alpha^\oplus \ni_Y$.

(PA_{*}) [Strict Point Axiom] For all relations $\rho : I \rightarrow X$ the identity $\rho = \sqcup_{x \in \rho} x$ holds.

(AC) [Relational Axiom of Choice] For all relations $\alpha : X \rightarrow Y$ there exists a univalent relation $f : X \rightarrow Y$ such that $f \sqsubseteq \alpha$ and $\text{dom}(f) = \text{dom}(\alpha)$.

It is worth to remark that almost all properties on relations stated here are also given in [11] with a subtly different fashion.

Let X and Y be a pair of objects in Dedekind category. By the rationality (RAT) there exists a pair of functions $p : R \rightarrow X$ and $q : R \rightarrow Y$ such that $p^\sharp q = \nabla_{XY}$ and $pp^\sharp \sqcap qq^\sharp = \text{id}_R$. The common domain R of p and q is called the *relational product* of X and Y , and will be denoted by $X \times Y$. Also the pair of functions p and q is called a *pair of projections* for X and Y . For relations $\rho : V \rightarrow X$ and $\sigma : V \rightarrow Y$ we define the pairing relation $\rho \top \sigma : V \rightarrow X \times Y$ by

$$\rho \top \sigma = \rho p^\sharp \sqcap \sigma q^\sharp.$$

In general $(\rho \top \sigma)p = \rho$ and $(\rho \top \sigma)q = \sigma$ do not hold. However they hold if ρ and σ are total, respectively. The composition $(\rho \top \sigma)\mu : V \rightarrow Z$ of $\rho \top \sigma : V \rightarrow X \times Y$ followed by a relation $\mu : X \times Y \rightarrow Z$ will often be denoted by $\rho \top_\mu \sigma$. Moreover, let $p_* : V \times W \rightarrow V$ and $q_* : V \times W \rightarrow W$ be a pair of projections for V and W . Then for relations $\kappa : V \rightarrow X$ and $\eta : W \rightarrow Y$, the product relation $\kappa \times \eta : V \times W \rightarrow X \times Y$ is defined by $\kappa \times \eta = p_* \kappa \top q_* \eta$. For the sharpness of pairing and product relations refer to [11, 6].

Proposition 1. *Let $f : V \rightarrow X$ and $g : W \rightarrow Y$ be functions, $\alpha : X \rightarrow Y$ and $\mu : X \times Y \rightarrow Y$ relations. Then*

- (a) $p^\sharp(p\alpha \sqcap q) = \alpha$ and $pp^\sharp(\mu \sqcap q) \sqcap q = \mu \sqcap q$,
- (b) $[p^\sharp(\mu \sqcap q)]^- = p^\sharp(\mu^- \sqcap q)$,
- (c) $p_*^\sharp[(f \times g)\mu g^\sharp \sqcap q_*] = f p^\sharp(\mu \sqcap q) g^\sharp$. $\square$

Lemma 1. *Let $f : V \rightarrow X$ and $g : V \rightarrow Y$ be functions, and $\delta : X \rightarrow Z$, $\mu : X \times Y \rightarrow Z$ relations in a Dedekind category. Then the following inclusion holds.*

$$f\delta \sqcap (f \top_\mu g) \sqsubseteq g(\nabla_{YZ} \delta^\sharp \top_\mu \text{id}_Y)$$

Proof.

$$\begin{aligned}
 f\delta \sqcap (f\top_\mu g) &\sqsubseteq [f(\delta\mu^\sharp p \sqcap \text{id}_X)p^\sharp \sqcap gq^\sharp]\mu \{ \text{DF} \} \\
 &= [f(p^\sharp\mu\delta^\sharp \sqcap \text{id}_X)p^\sharp \sqcap gq^\sharp]\mu \{ u^\sharp = u \text{ if } u \sqsubseteq \text{id}_X \} \\
 &\sqsubseteq (g\nabla_{YZ}\delta^\sharp p^\sharp \sqcap gq^\sharp)\mu \{ f p^\sharp \mu \sqsubseteq \nabla_{VZ} = g\nabla_{YZ} \} \\
 &= g(\nabla_{YZ}\delta^\sharp \top_\mu \text{id}_Y).
 \end{aligned}$$

□

3 Lattices in Dedekind Categories

In this section we investigate some elementary properties on lattices in relational categories, as reviewing [5]. A relation $\xi : X \rightarrow X$ in a Dedekind category is called a (partial) *order* if it is reflexive ($\text{id}_X \sqsubseteq \xi$), transitive ($\xi\xi \sqsubseteq \xi$) and antisymmetric ($\xi \sqcap \xi^\sharp \sqsubseteq \text{id}_X$). An order ξ is *complete* if $\sup(\rho, \xi)$ is total (consequently, a function by Prop. 2.3 in [7]) for all relations $\rho : V \rightarrow X$. The following lemma is useful to verify if two parallel univalent relations into an ordered object are identical.

Lemma 2. *Let $f, g : V \rightarrow X$ be univalent relations and $\xi : X \rightarrow X$ an order. If $f\xi = g\xi$, then $f = g$.*

Proof. An inclusion $f \sqsubseteq g$ is direct from $f = f \sqcap g\xi \sqsubseteq gg^\sharp f \sqsubseteq g\xi^\sharp f^\sharp f \sqsubseteq g\xi^\sharp$ and $f \sqsubseteq g\xi \sqcap g\xi^\sharp = g(\xi \sqcap \xi^\sharp) = g$. The converse inclusion $g \sqsubseteq f$ follows by symmetry. □

Throughout the rest of the paper we assume that $p : X \times X \rightarrow X$ and $q : X \times X \rightarrow X$ is a pair of relational projections in a Dedekind or Schröder category.

Definition 3. Let $\xi : X \rightarrow X$ be an order in a Dedekind category. Define four (univalent) relations $1_X : I \rightarrow X$, $0_X : I \rightarrow X$, $\vee_X : X \times X \rightarrow X$ and $\wedge_X : X \times X \rightarrow X$ by $1_X = \sup(\nabla_{IX}, \xi)$, $0_X = \sup(\nabla_{IX}, \xi^\sharp)$, $\vee_X = \sup(p \sqcup q, \xi)$ and $\wedge_X = \sup(p \sqcup q, \xi^\sharp)$, respectively. □

The relations 1_X , 0_X , $\vee_X$ and $\wedge_X$ will be denoted by 1 , 0 , $\vee$ and $\wedge$, respectively, unless no confusion occurs. Remark that $1 = \sup(\emptyset_{IX}, \xi^\sharp) = \nabla_{IX} \odot \xi$, $1\xi = 1$ and $1\xi^\sharp = \nabla_{IX}$ if 1 is a function. Dually $0 = \sup(\emptyset_{IX}, \xi) = \nabla_{IX} \odot \xi^\sharp$, $0\xi^\sharp = 0$ and $0\xi = \nabla_{IX}$ if 0 is a function.

Proposition 2. *Let $\xi : X \rightarrow X$ be an order, $f, g, h : V \rightarrow X$ functions, and $\rho, \sigma, \tau : I \rightarrow X$ relations. If $\vee = \sup(p \sqcup q, \xi)$ is a function, then*

- (a) $\vee\xi = p\xi \sqcap q\xi$, $\xi = p^\sharp\vee = q^\sharp\vee$ and $p \sqcup q \sqsubseteq \vee\xi^\sharp$,
- (b) $\text{id}_X \top_\vee \text{id}_X = \text{id}_X$,
- (c) $q \top_\vee p = \vee$ and $\rho \top_\vee \sigma = \sigma \top_\vee \rho$,
- (d) $(f \top_\vee g) \top_\vee h = f \top_\vee (g \top_\vee h)$ and $(\rho \top_\vee \sigma) \top_\vee \tau = \rho \top_\vee (\sigma \top_\vee \tau)$,
- (e) $p\xi p^\sharp \sqcap q\xi q^\sharp \sqsubseteq \vee\xi\vee^\sharp$ and $\rho\xi^\sharp \top_\vee \sigma\xi^\sharp \sqsubseteq (\rho \top_\vee \sigma)\xi^\sharp$.

□

Corollary 1. *Let $\xi : X \rightarrow X$ be an order, $f, g, h : V \rightarrow X$ functions, and $\rho, \sigma, \tau : I \rightarrow X$ relations. If $\wedge = \sup(p \sqcup q, \xi^\sharp)$ is a function, then*

- (a) $\wedge \xi^\sharp = p \xi^\sharp \sqcap q \xi^\sharp$, $\xi^\sharp = p^\sharp \wedge = q^\sharp \wedge$ and $p \sqcup q \sqsubseteq \wedge \xi$,
- (b) $\text{id}_X \top_\wedge \text{id}_X = \text{id}_X$,
- (c) $q \top_\wedge p = \wedge$ and $\rho \top_\wedge \sigma = \sigma \top_\wedge \rho$,
- (d) $(f \top_\wedge g) \top_\wedge h = f \top_\wedge (g \top_\wedge h)$ and $(\rho \top_\wedge \sigma) \top_\wedge \tau = \rho \top_\wedge (\sigma \top_\wedge \tau)$,
- (e) $p \xi^\sharp p^\sharp \sqcap q \xi^\sharp q^\sharp \sqsubseteq \wedge \xi^\sharp \wedge^\sharp$ and $\rho \xi \top_\wedge \sigma \xi \sqsubseteq (\rho \top_\wedge \sigma) \xi$. □

Now we give a formal definition of lattices in Dedekind categories.

Definition 4. A *lattice* in a Dedekind category $\mathcal{D}$ is a pair (X, ξ) of an object X and a relation $\xi : X \rightarrow X$ such that both of $\vee = \sup(p \sqcup q, \xi)$ and $\wedge = \sup(p \sqcup q, \xi^\sharp)$ are functions. A lattice (X, ξ) is *bounded* if $1 = \sup(\nabla_{IX}, \xi)$ and $0 = \sup(\nabla_{IX}, \xi^\sharp)$ are functions and $1 \sqcap 0 = \emptyset_{IX}$. □

Proposition 3. *Let (X, ξ) be a bounded lattice in a Dedekind category. Then*

- (a) $p \top_\vee \wedge = p$ and $p \top_\wedge \vee = p$,
- (b) $\nabla_{XI} 0 \top_\vee \text{id}_X = \text{id}_X$ and $\nabla_{XI} 1 \top_\wedge \text{id}_X = \text{id}_X$. □

Example 1. Let X be an object in a Dedekind category. The *power order* $\Xi_X : \wp(X) \rightarrow \wp(X)$ is defined by $\Xi_X = \exists_X \odot \exists_X^\sharp$, where $\exists_X : \wp(X) \rightarrow X$ is the membership relation. It is easy to see $\text{id}_{\wp(X)} \sqsubseteq \Xi_X$ and $\Xi_X \Xi_X \sqsubseteq \Xi_X$. The antisymmetry $\Xi_X \sqcap \Xi_X^\sharp \sqsubseteq \text{id}_{\wp(X)}$ follows from the rationality (RAT). Thus Ξ_X is in fact an order on $\wp(X)$. The power order Ξ_X is complete [4] and $(\wp(X), \Xi_X)$ is a distributive lattice by the following identities: Let $\rho : V \rightarrow \wp(X)$ be a relation, $f, g, h : V \rightarrow \wp(X)$ functions and $P, Q : \wp(X) \times \wp(X) \rightarrow \wp(X)$ a pair of projections. Then one can define two functions $\cup, \cap : \wp(X) \times \wp(X) \rightarrow \wp(X)$ by

$$\cup = \sup(P \sqcup Q, \Xi_X) \quad \text{and} \quad \cap = \inf(P \sqcup Q, \Xi_X)$$

and the following holds:

- (a) $\sup(\rho, \Xi_X) = (\rho \exists_X)^\circledast$ and $\sup(\rho, \Xi_X^\sharp) = (\rho \odot \exists_X)^\circledast$,
- (b) $1_{\wp(X)} = \nabla_{IX}^\circledast$ and $0_{\wp(X)} = \emptyset_{IX}^\circledast$,
- (c) $\cup \exists_X = P \exists_X \sqcup Q \exists_X$ and $\cap \exists_X = P \exists_X \sqcap Q \exists_X$,
- (d) $(f \top_\cup g) \exists_X = f \exists_X \sqcup g \exists_X$ and $(f \top_\cap g) \exists_X = f \exists_X \sqcap g \exists_X$,
- (e) $(f \top_\cup g) \top_\cap h = (f \top_\cap h) \top_\cup (g \top_\cap h)$. □

4 Boolean Algebras

In this section we re-formulate the concept of Boolean algebras in Dedekind categories.

Definition 5. A *Boolean algebra* in a Dedekind category $\mathcal{D}$ is a triple $(X, \xi, \neg)$ of an object X , a relation $\xi : X \rightarrow X$ and a function $\neg : X \rightarrow X$ such that

- (a) (X, ξ) is a bounded lattice,
- (b) $\text{id}_X \top_\vee \neg = \nabla_{XI} 1$,

- (c) $\text{id}_X \top_\wedge \neg = \nabla_{XI} 0$,
 (d) $(f \top_\vee g) \top_\wedge h = (f \top_\wedge h) \top_\vee (g \top_\wedge h)$ for all functions $f, g, h : V \rightarrow X$. $\square$

Note that Schmidt [11] also introduced the notion of Boolean lattices with relation methods, but his definition inspired by the matrix representation of Boolean orders is quite different from ours. The following are some fundamental properties of Boolean algebras in Dedekind categories.

Lemma 3. *Let $(X, \xi, \neg)$ be a Boolean algebra and $f, g : V \rightarrow X$ functions in a Dedekind category. If $f \top_\vee g = \nabla_{VI} 1$ and $f \top_\wedge g = \nabla_{VI} 0$, then $f = g \neg$.*

Proof.

$$\begin{array}{ll}
 f = f(\text{id}_X \top_\wedge \nabla_{XI} 1) & \{ \text{id}_X = \text{id}_X \top_\wedge \nabla_{XI} 1 \} \\
 = f \top_\wedge g \nabla_{XI} 1 & \{ f \nabla_{XI} = \nabla_{VI} = g \nabla_{XI} \} \\
 = f \top_\wedge g(\text{id}_X \top_\vee \neg) & \{ \nabla_{XI} 1 = \text{id}_X \top_\vee \neg \} \\
 = f \top_\wedge (g \top_\vee g \neg) & \{ g : \text{function} \} \\
 = (f \top_\wedge g) \top_\vee (f \top_\wedge g \neg) & \{ \text{Def. 5(d)} \} \\
 = (g \top_\wedge g \neg) \top_\vee (f \top_\wedge g \neg) & \{ f \top_\wedge g = \nabla_{VI} 0 = g \top_\wedge g \neg \} \\
 = (g \top_\vee f) \top_\wedge g \neg & \{ \text{Def. 5(d)} \} \\
 = \nabla_{VI} 1 \top_\wedge g \neg & \{ f \top_\vee g = \nabla_{VI} 1 \} \\
 = g \neg (\nabla_{XI} 1 \top_\wedge \text{id}_X) & \{ \nabla_{VI} = g \neg \nabla_{XI} \} \\
 = g \neg. & \{ \nabla_{XI} 1 \top_\wedge \text{id}_X = \text{id}_X \}
 \end{array}$$

$\square$

Proposition 4. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. Then*

- (a) $\xi^\# \sqcap \neg \xi^\# = \nabla_{XI} 0$, $(x \geq y) \wedge (\neg x \geq y) \rightarrow (y = 0)$
 (b) $\xi \sqcap \xi \neg = 0^\# \nabla_{IX}$, $(x \leq y) \wedge (x \leq \neg y) \rightarrow (x = 0)$
 (c) $\nabla_{XI} 0^- \sqcap \neg \xi^\# \sqsubseteq \xi^\# -$. $(y \neq 0) \wedge (\neg x \geq y) \rightarrow (x \not\geq y)$

$\square$

Corollary 2. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. Then*

- (a) $\text{id}_X = \neg \neg$ and $\neg^\# = \neg$, $(x = \neg \neg x)$
 (b) $0 \neg = 1$ and $1 \neg = 0$, $(\neg 0 = 1, \neg 1 = 0)$
 (c) $(\neg \times \neg) \wedge = \vee \neg$, (de Morgan's law) $(\neg x \wedge \neg y) = \neg(x \vee y)$
 (d) $\xi = \neg \xi^\# \neg$, $(x \leq y) \leftrightarrow (\neg x \geq \neg y)$
 (e) $\xi \neg = p^\#(\wedge 0^\# \nabla_{IV} \sqcap q)$, $(x \leq y) \leftrightarrow (x \wedge y = 0)$
 (f) $f \xi \neg g^\# = p_*^\#[(f \times g) \wedge 0^\# \nabla_{IV} \sqcap q_*]$,
 (g) $f \xi \neg g^\# = p_*^\#[(f \times g \neg) \wedge 0^- \neg \nabla_{IV} \sqcap q_*]$.

$\square$

Example 2. In Example 1 we have seen that $(\wp(X), \Xi_X)$ is a complete and distributive lattice for every object X in a Dedekind category with membership relations. Let X be an object in a Schröder category with membership relations. Then the complement function $\neg_{\wp(X)} : \wp(X) \rightarrow \wp(X)$ is defined by

$$\neg_{\wp(X)} \ni x = \ni x^-, \quad \text{i. e.} \quad \neg_{\wp(X)} = (\ni x^-)^\oplus.$$

Using the identities obtained in Example 1, it is easy to verify $\text{id}_{\wp(X)} \top_{\neg_{\wp(X)}} = \nabla_{\wp(X)I} 1_{\wp(X)}$ and $\text{id}_{\wp(X)} \top_{\neg_{\wp(X)}} = \nabla_{\wp(X)I} 0_{\wp(X)}$. Therefore the power object $\wp(X) = (\wp(X), \Xi_X, \neg_{\wp(X)})$ is a (complete) Boolean algebra. $\square$

5 Homomorphisms

In this section we state the definition of homomorphisms of Boolean algebras and the basic properties of homomorphisms.

Definition 6. Let $(X, \xi_X, \neg_X)$ and $(Y, \xi_Y, \neg_Y)$ be Boolean algebras in a Dedekind category. A function $f : X \rightarrow Y$ is called a *homomorphism* of Boolean algebras if $\neg_X f = f \neg_Y$ and $\vee_X f = (f \times f) \vee_Y$. $\square$

The following proposition shows the fundamental properties on homomorphisms of Boolean algebras.

Proposition 5. Let $(X, \xi_X, \neg_X)$ and $(Y, \xi_Y, \neg_Y)$ be Boolean algebras in a Dedekind category. If $f : X \rightarrow Y$ is a homomorphism of Boolean algebras, then the following holds.

- (a) $\wedge_X f = (f \times f) \wedge_Y$,
- (b) $0_X f = 0_Y$,
- (c) $1_X f = 1_Y$,
- (d) $\xi_X f \sqsubseteq f \xi_Y$,
- (e) If $fg = \text{id}_X$ and $gf = \text{id}_Y$, then g is a homomorphism. $\square$

The following theorem gives a sufficient condition for a homomorphisms of Boolean algebras to be injective. This result will be used to demonstrate that Stone maps (discussed in Section 6) are injective.

Theorem 1. Let $(X, \xi_X, \neg_X)$ and $(Y, \xi_Y, \neg_Y)$ be Boolean algebras. A homomorphism $f : X \rightarrow Y$ is injective iff $0_Y f^\# = 0_X$.

Proof. First assume that f is injective. Since $0_X f = 0_Y$ by Prop. 5(c), identities $0_Y f^\# = 0_X f f^\# = 0_X \text{id}_X = 0_X$ hold. Conversely assume $0_Y f^\# = 0_X$. Then we have

$$\begin{aligned}
 f \xi_Y f^\# &= p^\#[(f \times f \neg_Y) \wedge_Y 0_Y^\# \nabla_{IX} \sqcap q] && \{ \text{Cor. 2(f)} \} \\
 &= p^\#[(f \times \neg_X f) \wedge_Y 0_Y^\# \nabla_{IX} \sqcap q] && \{ f \neg_Y = \neg_X f \} \\
 &= p^\#[(\text{id}_X \times \neg_X)(f \times f) \wedge_Y 0_Y^\# \nabla_{IX} \sqcap q] \\
 &= p^\#[(\text{id}_X \times \neg_X) \wedge_X f 0_Y^\# \nabla_{IX} \sqcap q] && \{ (f \times f) \wedge_Y = \wedge_X f \} \\
 &= p^\#[(\text{id}_X \times \neg_X) \wedge_X 0_X^\# \nabla_{IX} \sqcap q] && \{ 0_Y f^\# = 0_X \} \\
 &= \xi_X, && \{ \text{Cor. 2(f)} \}
 \end{aligned}$$

and so

$$\begin{aligned}
 f f^\# &= f(\xi_Y \sqcap \xi_Y^\#) f^\# && \{ \text{id}_Y = \xi_Y \sqcap \xi_Y^\# \} \\
 &= f \xi_Y f^\# \sqcap (f \xi_Y f^\#)^\# && \{ f : \text{function} \} \\
 &= \xi_X \sqcap \xi_X^\# && \{ f \xi_Y f^\# = \xi_X \} \\
 &= \text{id}_X. && \{ \xi_X \sqcap \xi_X^\# = \text{id}_X \}
 \end{aligned}$$

$\square$

6 Filters

In the section we re-formulate some concepts on filters in relational categories, which are indispensable to establish Stone's theorem for Boolean algebras.

Definition 7. Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Dedekind category. A relation $\rho : V \rightarrow X$ is called a (*proper*) *filter* on X if

(a) $\nabla_{VI}1 \sqsubseteq \rho$,	$\{ 1 \in \rho \}$
(b) $\rho 0^\# = \emptyset_{VI}$,	$\{ 0 \notin \rho \}$
(c) $\rho \xi \sqsubseteq \rho$,	$\{ x \in \rho, x \leq y \rightarrow y \in \rho \}$
(d) $\rho \top_\wedge \rho \sqsubseteq \rho$.	$\{ x, y \in \rho \rightarrow x \wedge y \in \rho \}$
-

It is trivial that the condition (d) above is equivalent to $\rho \sqcap \nabla_{VI}0 = \emptyset_{VX}$. Every filter $\rho : V \rightarrow X$ is total, since it contains a total relation $\nabla_{VI}1$ by Def. 7(a). Note that $\rho \top \rho = \rho \wedge^\#$ iff $\rho \xi \sqsubseteq \rho$ and $\rho \top_\wedge \rho \sqsubseteq \rho$.

Proposition 6. Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Dedekind category. If $\rho : V \rightarrow X$ is a filter on X , then $\rho \sqcap \rho \neg = \emptyset_{VX}$.

Proof. Let $\rho : V \rightarrow X$ be a filter. By the relational axiom of choice (AC) there is a univalent relation $f : V \rightarrow X$ such that $f \sqsubseteq \rho \sqcap \rho \neg$ and $\text{dom}(f) = \text{dom}(\rho \sqcap \rho \neg)$. Thus we have

$$\begin{aligned}
 f \nabla_{XI}0 &= f \nabla_{XI}0 \sqcap \nabla_{VI}0 && \{ f \nabla_{XI} \sqsubseteq \nabla_{VI} \} \\
 &= f(\text{id}_X \top_\wedge \neg) \sqcap \nabla_{VI}0 && \{ \nabla_{XI}0 = \text{id}_X \top_\wedge \neg \} \\
 &= (f \top_\wedge f \neg) \sqcap \nabla_{VI}0 && \{ f : \text{univalent} \} \\
 &\sqsubseteq (\rho \top_\wedge \rho) \sqcap \nabla_{VI}0 && \{ f \sqsubseteq \rho \sqcap \rho \neg \} \\
 &\sqsubseteq \rho \sqcap \nabla_{VI}0 && \{ \text{Def. 7(d)} \} \\
 &\sqsubseteq (\rho 0^\# \sqcap \nabla_{VI})0 && \{ \text{DF} \} \\
 &= \emptyset_{VX}, && \{ \text{Def. 7(b)} \}
 \end{aligned}$$

and so $f \sqsubseteq f \nabla_{XI}0 0^\# \nabla_{IX} = \emptyset_{VX}$, which shows $\rho \sqcap \rho \neg = \emptyset_{VX}$. □

Definition 8. Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Dedekind category.

- (a) A filter $\rho : V \rightarrow X$ is *prime* if $\rho \vee^\# \sqsubseteq \rho(p \sqcup q)^\#$.
- (b) A filter $\rho : V \rightarrow X$ is *ultra* if $\rho \sqcup \rho \neg = \nabla_{VX}$.
- (c) A filter $\rho : V \rightarrow X$ is *maximal* if it is maximal among filters. □

Note that $\rho(p \sqcup q)^\# = \rho \vee^\#$ iff $\rho \xi \sqsubseteq \rho$ and $\rho \vee^\# \sqsubseteq \rho(p \sqcup q)^\#$.

Proposition 7. Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Dedekind category. For all objects V the ordered set $\mathcal{F}(V, X)$ of all filters $\rho : V \rightarrow X$ on X is inductive, that is, every nonempty chain in $\mathcal{F}(V, X)$ has an upper bound.

Proof. It is trivial that $\mathcal{F}(V, X)$ is ordered by the inclusion $\sqsubseteq$ of relations. Let $\{\rho_\lambda \mid \lambda \in \Lambda\}$ be a nonempty chain in $\mathcal{F}(V, X)$ and set $\rho = \sqcup_{\lambda \in \Lambda} \rho_\lambda$. Then ρ is also a filter on X as follows. Three conditions $\nabla_{VI} 1 \sqsubseteq \rho$, $\rho \sqcap \nabla_{VI} 0 = \emptyset_{VX}$ and $\rho \xi \sqsubseteq \rho$ are trivial. The condition $\rho \top_\wedge \rho \sqsubseteq \rho$ simply follows from

$$\begin{aligned} \rho \top_\wedge \rho &= [(\sqcup_\lambda \rho_\lambda) p^\# \sqcap (\sqcup_\lambda \rho_\lambda) q^\#] \wedge \\ &= \sqcup_{\lambda, \lambda'} (\rho_\lambda p^\# \sqcap \rho_{\lambda'} q^\#) \wedge \\ &= \sqcup_\lambda (\rho_\lambda p^\# \sqcap \rho_\lambda q^\#) \wedge \quad \{ \{\rho_\lambda \mid \lambda \in \Lambda\} : \text{chain} \} \\ &= \sqcup_\lambda (\rho_\lambda \top_\wedge \rho_\lambda) \\ &\sqsubseteq \sqcup_\lambda \rho_\lambda. \quad \{ \rho_\lambda : \text{filter} \} \end{aligned}$$

This completes the proof. $\square$

Proposition 8. *Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Dedekind category. If $\rho : I \rightarrow X$ is a filter on X and $x : I \rightarrow X$ is an I -point, then $\rho_x = (\rho \top_\wedge x) \xi$ satisfies $\rho \sqcup a \sqsubseteq \rho_x$, $\rho_x \xi \sqsubseteq \rho_x$ and $\rho_x \top_\wedge \rho_x \sqsubseteq \rho_x$.* $\square$

Remark that the relation ρ_x discussed above does not always satisfies the condition $\rho_x 0^\# = \emptyset_{IX}$.

Theorem 2. *Let $X = (X, \xi, \neg)$ be a Boolean algebra in a Schröder category.*

- (a) *A filter $\rho : V \rightarrow X$ is a ultra filter iff it is a prime filter.*
- (b) *Every ultra filter $\rho : V \rightarrow X$ is a maximal filter.*
- (c) *Every maximal filter $\rho : I \rightarrow X$ is a ultra filter.*

Proof. (a) First assume that $\rho : V \rightarrow X$ is a ultra filter. Then we have

$$\begin{aligned} \rho \vee^\# &= \rho \neg \neg \vee^\# & \{ \text{id}_X = \neg \neg \} \\ &= \rho \neg \wedge^\# (\neg \times \neg) & \{ \text{de Morgan} \} \\ &= \rho^- \wedge^\# (\neg \times \neg) & \{ \rho \neg = \rho^- \} \\ &= (\rho \wedge^\#)^- (\neg \times \neg) & \{ \wedge : \text{function} \} \\ &= (\rho p^\# \sqcap \rho q^\#)^- (\neg \times \neg) & \{ \rho \wedge^\# = \rho \top_\rho \} \\ &= \rho^- (p^\# \sqcup q^\#) (\neg \times \neg) & \{ \text{de Morgan} \} \\ &= \rho \neg (\neg p^\# \sqcup \neg q^\#) & \{ \rho^- = \rho \neg, (\neg \times \neg) p = p \neg \} \\ &= \rho (p \sqcup q)^\#, & \{ \neg \neg = \text{id}_X \} \end{aligned}$$

which shows that ρ is a prime filter. Conversely assume that $\rho : V \rightarrow X$ is a prime filter. Then

$$\begin{aligned} \rho^- &= \rho^- \neg \neg & \{ \text{id}_X = \neg \neg \} \\ &= \rho^- (\text{id}_X \top_\neg) q \neg & \{ \neg = (\text{id}_X \top_\neg) q \} \\ &= [\rho^- (\text{id}_X \top_\neg) \sqcap \rho^- p^\#] q \neg & \{ \text{id}_X \top_\neg \sqsubseteq p^\# \} \\ &\sqsubseteq [\rho^- (\text{id}_X \top_\neg) \vee \vee^\# \sqcap \rho^- p^\#] q \neg & \{ \vee : \text{total} \} \\ &= (\rho^- \nabla_{XI} 1 \vee^\# \sqcap \rho^- p^\#) q \neg & \{ \text{id}_X \top_\vee \neg = \nabla_{XI} 1 \} \\ &\sqsubseteq (\nabla_{VI} 1 \vee^\# \sqcap \rho^- p^\#) q \neg & \{ \rho^- \nabla_{XI} \sqsubseteq \nabla_{VI} \} \\ &\sqsubseteq (\rho \vee^\# \sqcap \rho^- p^\#) q \neg & \{ \nabla_{VI} 1 \sqsubseteq \rho \} \\ &\sqsubseteq [\rho (p^\# \sqcup q^\#) \sqcap \rho^- p^\#] q \neg & \{ \rho : \text{prime} \} \\ &\sqsubseteq \rho \neg. & \{ \rho^- p^\# = (\rho p^\#)^- \} \end{aligned}$$

which implies that $\rho \sqcup \rho^\neg = \nabla_{VX}$.

(b) Let $\rho : V \rightarrow X$ be a ultra filter and σ a filter with $\rho \sqsubseteq \sigma$. Then we have

$$\begin{aligned} \sigma &= \sigma \sqcap (\rho \sqcup \rho^\neg) & \{ \rho \sqcup \rho^\neg = \nabla_{VX} \} \\ &= (\sigma \sqcap \rho) \sqcup (\sigma \sqcap \rho^\neg) \\ &\sqsubseteq \rho \sqcup (\sigma \sqcap \sigma^\neg) & \{ \rho \sqsubseteq \sigma \} \\ &= \rho, & \{ \text{Prop. 6.1: } \sigma \sqcap \sigma^\neg = \emptyset_{VX} \} \end{aligned}$$

which proves $\sigma = \rho$. Hence ρ is maximal.

(c) Let $\rho : I \rightarrow X$ be a maximal filter and set $\sigma = (\rho \sqcup \rho^\neg)^\neg$. We have to see $\sigma = \emptyset_{IX}$. Assume $\sigma \neq \emptyset_{IX}$. Then by the strict point axiom (PA_{*}) there exists an I -point $x : I \rightarrow X$ such that $x \sqsubseteq \sigma$. Set $\rho_x = (\rho \top_\wedge x)\xi$.

(i) In the case that $\rho_x \sqcap 0 = \emptyset_{IX}$: Then ρ_x is a filter by Prop. 6.2 and $\rho = \rho_x$ holds by the maximality of ρ . Hence we have

$$x \sqsubseteq \rho \sqcap \sigma \sqsubseteq \rho \sqcap \rho^\neg = \emptyset_{IX},$$

which contradicts $\text{id}_I \neq \emptyset_{II}$.

(ii) In the case that $\rho_x \sqcap 0 \neq \emptyset_{IX}$: Again by (PA_{*}) we have $0 \sqsubseteq \rho_x$ and so $0 \sqsubseteq \rho \top_\wedge x$, because $0 \sqsubseteq \eta\xi$ implies $0 = 00^\#0 \sqsubseteq \eta\xi0^\#0 \sqsubseteq \eta0^\#0 \sqsubseteq \eta$ by $0\xi^\# = 0$. Hence it holds that

$$\begin{aligned} x &= 00^\#\nabla_{IX} \sqcap x & \{ 0 : \text{total} \} \\ &\sqsubseteq (\rho \top x) \wedge 0^\#\nabla_{IX} \sqcap x & \{ 0 \sqsubseteq (\rho \top x) \wedge \} \\ &\sqsubseteq (\rho \top x) [\wedge 0^\#\nabla_{IX} \sqcap (\rho \top x)^\#x] & \{ \text{DF} \} \\ &\sqsubseteq \rho p^\# (\wedge 0^\#\nabla_{IX} \sqcap q) & \{ \rho \top x = \rho p^\# \sqcap x q^\# \} \\ &= \rho \xi^\neg & \{ \text{Cor. 2.2(e)} \} \\ &\sqsubseteq \rho^\neg, & \{ \rho : \text{filter} \} \end{aligned}$$

and

$$\begin{aligned} x &\sqsubseteq \rho^\neg \sqcap \sigma & \{ x \sqsubseteq \rho^\neg, x \sqsubseteq \sigma \} \\ &\sqsubseteq \rho^\neg \sqcap \rho^\neg{}^\neg & \{ \sigma \sqsubseteq (\rho^\neg)^\neg \} \\ &= (\rho \sqcap \rho^\neg)^\neg \\ &= \emptyset_{IX}, \end{aligned}$$

which is also a contradiction. Therefore we have proved $\rho \sqcup \rho^\neg = \nabla_{IX}$. $\square$

7 Stone Maps

Let $(X, \xi, \neg)$ be a Boolean algebra in a Dedekind category. By the rationality (RAT) of relations there exists an injection $j : U \rightarrow \wp(X)$ such that

$$\begin{aligned} j^\# j &= \text{id}_{\wp(X)} \sqcap \nabla_{\wp(X)I} 1 \exists_X^\# \sqcap (\exists_X 0^\# \ominus \emptyset_{I\wp(X)}) \sqcap (\exists_X \xi \ominus \exists_X^\#) \\ &\sqcap [(\exists_X \top_\wedge \exists_X) \ominus \exists_X^\#] \sqcap [\exists_X \vee^\# \ominus (\rho \sqcup q) \exists_X^\#]. \end{aligned}$$

The injection $j : U \rightarrow \wp(X)$ will be called the *primal injection* of X . The object U is an extension of the set of all prime filters on X .

Proposition 9. *Let $(X, \xi, \neg)$ be a Boolean algebra and $f : V \rightarrow \wp(X)$ a function in a Dedekind category. Then $f \ni_X : V \rightarrow X$ is a prime filter iff $f^\# f \sqsubseteq j^\# j$.*

Proof. The statement is obvious from the following equivalences.

- (a) $\nabla_{VI} 1 \sqsubseteq f \ni_X \leftrightarrow f^\# f \sqsubseteq \nabla_{\wp(X)I} 1 \ni_X^\#$,
- (b) $f \ni_X 0^\# = \emptyset_{VI} \leftrightarrow f^\# f \sqsubseteq \ni_X 0^\# \odot \emptyset_{I\wp(X)}$,
- (c) $f \ni_X \xi \sqsubseteq f \ni_X \leftrightarrow f^\# f \sqsubseteq \ni_X \xi \odot \ni_X^\#$,
- (d) $f \ni_X \top_\wedge f \ni_X \sqsubseteq f \ni_X \leftrightarrow f^\# f \sqsubseteq (\ni_X \top_\wedge \ni_X) \odot \ni_X^\#$,
- (e) $f \ni_X \vee^\# \sqsubseteq f \ni_X (p \sqcup q)^\# \leftrightarrow f^\# f \sqsubseteq \ni_X \vee^\# \odot (p \sqcup q) \ni_X^\#$. □

In particular the relation $j \ni_X$ is a prime filter by the last proposition.

Proposition 10. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. Then the primal injection $j : U \rightarrow \wp(X)$ satisfies $\nabla_{IU} j \ni_X = 0^-$.*

Proof. The inclusion $\nabla_{IU} j \ni_X \sqsubseteq 0^-$ is direct from

$$\begin{aligned} 0 \sqcap \nabla_{IU} j \ni_X &\sqsubseteq (0 \ni_X^\# j^\# \sqcap \nabla_{IU}) j \ni_X \{ \text{DF} \} \\ &= \emptyset_{IX}. \quad \{ j \ni_X 0^\# = \emptyset_{UI} \} \end{aligned}$$

We now show the converse inclusion $0^- \sqsubseteq \nabla_{IU} j \ni_X$. By the point axiom (AP_{*}) $\sqcup_{x \in 0^-} x = 0^-$ holds. Let $x : I \rightarrow X$ be an I -point with $x \in 0^-$. Then it is easy to see that $x\xi : I \rightarrow X$ is a filter on X . Recall the ordered set $\mathcal{F}(I, X)$ is inductive set by means of Prop. 7. Therefore by Zorn's lemma (in set theory) there is a maximal filter $\hat{\rho} : I \rightarrow X$ such that $\rho \sqsubseteq \hat{\rho}$. By virtue of Theorem 2 $\hat{\rho}$ is a prime filter and so $\hat{\rho}^\# \hat{\rho} \sqsubseteq j^\# j$ by the last proposition. Therefore we have $x \sqsubseteq \hat{\rho}^\# \ni_X \hat{\rho}^\# j^\# j \ni_X \sqsubseteq \nabla_{IU} j \ni_X$. This implies $0^- = \sqcup_{x \in 0^-} x \sqsubseteq \nabla_{IU} j \ni_X$, which completes the proof. □

For a Boolean algebra $(X, \xi, \neg)$ in a Dedekind category we define the Stone map $S : X \rightarrow \wp(U)$ to be a unique function such that $S \ni_U = \ni_X^\# j^\#$, that is, $S = (\ni_X^\# j^\#)^\circ$. $xS = \{ \rho : \text{prime filter} \mid x \in \rho \}$

Proposition 11. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. The Stone map $S : X \rightarrow \wp(U)$ is a homomorphism of Boolean algebras.*

Proof. $\vee S = (S \times S) \cup_{\wp(U)} :$

$$\begin{aligned} \vee S \ni_U &= \vee \ni_X^\# j^\# & \{ S \ni_U = \ni_X^\# j^\# \} \\ &= (p \sqcup q) \ni_X^\# j^\# & \{ \text{Prop. 9(c)} : j \ni_X \vee^\# = j \ni_X (p \sqcup q)^\# \} \\ &= (p \sqcup q) S \ni_U & \{ \ni_X^\# j^\# = S \ni_U \} \\ &= (S \times S) (P \sqcup Q) \ni_U \\ &= (S \times S) \cup_{\wp(U)} \ni_U. \quad \{ (P \sqcup Q) \ni_U = \cup_{\wp(U)} \ni_U \} \end{aligned}$$

$\neg S = S \neg_{\wp(U)} :$

$$\begin{aligned} \neg S \ni_U &= \neg \ni_X^\# j^\# & \{ S \ni_U = \ni_X^\# j^\# \} \\ &= \ni_X^{-\#} j^\# & \{ (\text{U2}) : j \ni_X \neg = j \ni_X^- \} \\ &= (\ni_X^\# j^\#)^- & \{ j : \text{function} \} \\ &= (S \ni_U)^- & \{ \ni_X^\# j^\# = S \ni_U \} \\ &= S \ni_U^- & \{ S : \text{function} \} \\ &= S \neg_{\wp(U)} \ni_U. \quad \{ \ni_U^- = \neg_{\wp(U)} \ni_U \} \end{aligned}$$

□

Theorem 3 (Stone). *For every Boolean algebra $(X, \xi, \neg)$ in a Schröder category the Stone map $S : X \rightarrow \wp(U)$ is injective.*

Proof. It holds that

$$\begin{aligned} 0_{\wp(U)} S^\# &= (\nabla_{IU} \exists_U^\#)^- S^\# \{ 0_{\wp(U)} = (\nabla_{IU} \exists_U^\#)^- \} \\ &= (\nabla_{IU} \exists_U^\# S^\#)^- \{ S : \text{function} \} \\ &= (\nabla_{IU} j \exists_X)^- \{ S \exists_U = \exists_X^\# j^\# \} \\ &= 0. \quad \{ \text{Prop. 10: } \nabla_{IU} j \exists_X = 0^- \} \end{aligned}$$

Hence the Stone map S is injective by Theorem 11. $\square$

8 Atoms

In the final section we review the special representation of Boolean algebras by atoms.

Definition 9. Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. An injection $i : A \rightarrow X$ with $\nabla_{IA} i = 0^- \sqcap (0^- \xi_-)^-$ will be called the *atomic injection* of X , where $\xi_- = \xi \sqcap \text{id}_X^-$. $\square$

The atomic injection i does exist by the rationality (RAT) of relations.

Proposition 12. *Let $X = (X, \xi, \neg)$ be a Boolean algebra and $i : A \rightarrow X$ the atomic injection of X . Then the following holds.*

- (a) $(0^- \xi_-)^- = \nabla_{IX} \ominus (0^\# \nabla_{IX} \sqcup \xi_-^-)$,
- (b) $\nabla_{XA} i \sqsubseteq 0^\# \nabla_{IX} \sqcup \xi_-^-$,
- (c) $\nabla_{XA} i \sqcap 0^- \nabla_{IX} \sqcap \xi \sqsubseteq \text{id}_X$, $\{ (a \in A) \wedge (x \neq 0) \wedge (x \leq a) \rightarrow (x = a) \}$
- (d) $i \xi i^\# = \text{id}_A$, $\{ (a, b \in A) \rightarrow [(a \leq b) \leftrightarrow (a = b)] \}$
- (e) $\neg \xi^\# i^\# = \xi^\# \neg i^\#$.

Proof. (a) Set $\theta = 0^\# \nabla_{IX} \sqcup \xi_-^-$. First note that $0\theta = \nabla_{IX}$ (since $00^\# = \text{id}_I$) and

$$\begin{aligned} 0^- \theta^- &= 0^- ((0^\# \nabla_{IX})^- \sqcap \xi_-^-) \{ \text{de Morgan} \} \\ &= 0^- (0^- \nabla_{IX} \sqcap \xi_-^-) \{ \nabla_{XI} : \text{function} \} \\ &= 0^- \xi_-^-. \quad \{ \text{DF} \} \end{aligned}$$

Hence we have

$$\begin{aligned} \nabla_{IX} \ominus \theta &= (0 \sqcup 0^-) \ominus \theta \quad \{ \nabla_{IX} = 0 \sqcup 0^- \} \\ &= (0 \ominus \theta) \sqcap (0^- \ominus \theta) \\ &= 0\theta \sqcap (0^- \ominus \theta) \quad \{ 0 : \text{function} \} \\ &= 0^- \ominus \theta \quad \{ 0\theta = \nabla_{IX} \} \\ &= (0^- \theta^-)^- \\ &= (0^- \xi_-^-)^-. \quad \{ 0^- \theta^- = 0^- \xi_-^- \} \end{aligned}$$

(b) The inclusion $\nabla_{XA}i \sqsubseteq 0^\sharp \nabla_{IX} \sqcup \xi^- (= \theta)$ is direct from

$$\begin{aligned} \nabla_{XA}i &= \nabla_{XI} \nabla_{IA}i && \{ \nabla_{XA} = \nabla_{XI} \nabla_{IA} \} \\ &\sqsubseteq \nabla_{XI}(\nabla_{IX} \odot \theta) && \{ (a) \} \\ &= \nabla_{XI} \nabla_{IX} \odot \theta && \{ \nabla_{XI} : \text{function} \} \\ &= \nabla_{XX} \odot \theta && \{ \nabla_{XI} \nabla_{IX} = \nabla_{XX} \} \\ &\sqsubseteq \text{id}_X \odot \theta && \{ \text{id}_X \sqsubseteq \nabla_{XX} \} \\ &= \theta. \end{aligned}$$

(c) The inclusion $\nabla_{XA}i \sqcap 0^{-\sharp} \nabla_{IX} \sqcap \xi \sqsubseteq \text{id}_X$ is trivial from

$$\begin{aligned} &\nabla_{XA}i \sqcap 0^{-\sharp} \nabla_{IX} \sqcap \xi \\ &\sqsubseteq (0^\sharp \nabla_{IX} \sqcup \xi^- \sqcup \text{id}_X) \sqcap 0^{-\sharp} \nabla_{IX} \sqcap \xi \{ (b) \} \\ &\sqsubseteq \text{id}_X. \end{aligned}$$

(d) First $\text{id}_A = ii^\sharp \sqsubseteq i\xi i^\sharp$ is trivial. The converse inclusion is deduced by

$$\begin{aligned} i\xi i^\sharp &= i(i^\sharp \nabla_{AA}i \sqcap \xi)i^\sharp && \{ \text{DF} \} \\ &\sqsubseteq i(\nabla_{XA}i \sqcap i^\sharp \nabla_{AX} \sqcap \xi)i^\sharp && \{ i^\sharp \nabla_{AA} \sqsubseteq \nabla_{XA}, \nabla_{AA}i \sqsubseteq \nabla_{AX} \} \\ &\sqsubseteq i(\nabla_{XA}i \sqcap 0^{-\sharp} \nabla_{IX} \sqcap \xi)i^\sharp && \{ \nabla_{XA}i \sqsubseteq \nabla_{XI} 0^- \} \\ &\sqsubseteq ii^\sharp && \{ (c) \} \\ &= \text{id}_A. && \{ i : \text{injection} \} \end{aligned}$$

(e) The inclusion $\neg \xi^\sharp i^\sharp \sqsubseteq \xi^\sharp \neg i^\sharp$ is immediate from

$$\begin{aligned} \neg \xi^\sharp i^\sharp &= (\nabla_{XA}i \sqcap \neg \xi^\sharp)i^\sharp && \{ \text{DF} \} \\ &\sqsubseteq (\nabla_{XI} 0^- \sqcap \neg \xi^\sharp)i^\sharp && \{ \nabla_{XA}i \sqsubseteq \nabla_{XI} 0^- \} \\ &\sqsubseteq \xi^\sharp \neg i^\sharp. && \{ \text{Prop. 4(d)} \} \end{aligned}$$

To see the converse inclusion $\xi^\sharp \neg i^\sharp \sqsubseteq \neg \xi^\sharp i^\sharp$ we first note

$$(A) \quad \xi^-^\sharp = (\neg \xi \neg)^- = \neg \xi^- \neg = p^\sharp(\theta \sqcap q)$$

by Cor. 2(d) and 2(g), where $\theta = (\neg \times \text{id}_X) \wedge 0^{-\sharp} \nabla_{IX}$. Then we have

$$\begin{aligned} \xi^-^\sharp i^\sharp i &= \nabla_{XA}i \sqcap \xi^-^\sharp && \{ \text{DF} \} \\ &= \nabla_{XA}i \sqcap p^\sharp(\theta \sqcap q) && \{ (A) \} \\ &\sqsubseteq \nabla_{XA}i \sqcap p^\sharp(\neg \times \text{id}_X) \wedge (0^{-\sharp} \nabla_{IX} \sqcap \xi) && \{ q \sqsubseteq (\neg \times \text{id}_X) \wedge \xi \} \\ &\sqsubseteq p^\sharp(\neg \times \text{id}_X) \wedge (\nabla_{XA}i \sqcap 0^{-\sharp} \nabla_{IX} \sqcap \xi) && \{ \text{DF} \} \\ &\sqsubseteq p^\sharp(\neg \times \text{id}_X) \wedge && \{ \text{Prop. 12(c)} \} \\ &\sqsubseteq p^\sharp(\neg \times \text{id}_X) p \xi^\sharp && \{ \text{Prop. 2(g)} \} \\ &= p^\sharp p \neg \xi^\sharp \\ &\sqsubseteq \neg \xi^\sharp. \end{aligned}$$

This completes the proof. $\square$

For a Boolean algebra $(X, \xi, \neg)$ in a Schröder category we define a function $t : X \rightarrow \wp(A)$ by $t = (\xi^\sharp i^\sharp)^\circ$. $xt = x(\xi^\sharp i^\sharp)^\circ = \{a \in A \mid x \geq a\}$

Proposition 13. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. The function $t = (\xi^\# i^\#)^\circ : X \rightarrow \wp(A)$ is a homomorphism of Boolean algebras.*

Proof. $\wedge t = (t \times t) \cap_{\wp(A)} :$

$$\begin{aligned}
 \wedge t \ni_A &= \wedge \xi^\# i^\# & \{ t \ni_X &= \xi^\# i^\# \} \\
 &= (p \xi^\# \sqcap q \xi^\#) i^\# \\
 &= p \xi^\# i^\# \sqcap q \xi^\# i^\# \\
 &= p t \ni_A \sqcap q t \ni_A \\
 &= (t \times t) P \ni_A \sqcap (t \times t) Q \ni_A \\
 &= (t \times t) (P \ni_A \sqcap Q \ni_A) \\
 &= (t \times t) \cap_{\wp(A)} \ni_A, & \{ P \ni_A \sqcap Q \ni_A &= \cap_{\wp(A)} \ni_A \}
 \end{aligned}$$

(b) $\neg t = t \neg_{\wp(A)} :$

$$\begin{aligned}
 \neg t \ni_A &= \neg \xi^\# i^\# & \{ t \ni_A &= \xi^\# i^\# \} \\
 &= \xi^- \# i^\# & \{ \text{Prop. 12(f)} \} \\
 &= (\xi^\# i^\#)^- & \{ i : \text{function} \} \\
 &= (t \ni_A)^- & \{ \xi^\# i^\# = t \ni_A \} \\
 &= t \ni_A^- & \{ t : \text{function} \} \\
 &= t \neg_{\wp(A)} \ni_A. & \{ \ni_A^- = \neg_{\wp(A)} \ni_A \}
 \end{aligned}$$

Hence $\wedge t = (t \times t) \cap_{\wp(A)}$ and $\neg t = t \neg_{\wp(A)}$ follows by the extensionality of membership relations. $\square$

Definition 10. A Boolean algebra X is *atomic* if $\nabla_{IA} i \xi = 0^-$.

It is trivial that X is atomic iff $\nabla_{XA} i \xi = \nabla_{XI} 0^-$. ($\nabla_{XA} = \nabla_{XI} \nabla_{IX} \nabla_{XA} \sqsubseteq \nabla_{XI} \nabla_{IA}$ and $\nabla_{II} = \text{id}_I \sqsubseteq 0 0^\# \sqsubseteq \nabla_{IX} \nabla_{XI}$.)

Theorem 4. *Let $(X, \xi, \neg)$ be a Boolean algebra in a Schröder category. Then*

- (a) *t is injective iff X is atomic.*
- (b) *If ξ is complete, then t is surjective.*

Proof. (a) As $0_{\wp(A)} = (\nabla_{IA} \ni_A)^\#$ and $0_{\wp(A)}^\# t^\# = \nabla_{IA} i \xi$, it is clear that X is atomic iff $0^- = 0_{\wp(A)}^\# t^\#$ iff $0 = 0_{\wp(A)} t^\#$ iff t is injective by Theorem 11.

(b) As ξ is complete, $\text{sup}(\ni_A i, \xi)$ is a function. Thus we have

$$\begin{aligned}
 \text{sup}(\ni_A i, \xi) t \ni_A &= \text{sup}(\ni_A i, \xi) \xi^\# i^\# & \{ t \ni_A &= \xi^\# i^\# \} \\
 &= \text{sup}(\ni_A i, \xi) \xi^- \neg i^\# & \{ \text{Prop. 12(f)} \text{ and Cor. 2(d)} \} \\
 &= [\text{sup}(\ni_A i, \xi) \xi]^- \neg i^\# & \{ \text{sup}(\ni_A i, \xi) : \text{function} \} \\
 &= (\ni_A i \odot \xi)^- \neg i^\# & \{ \text{sup}(\rho, \xi) \xi = \rho \odot \xi \} \\
 &= \ni_A i \xi^- \neg i^\# & \{ \alpha \odot \beta = (\alpha \beta^-)^- \} \\
 &= \ni_A i \xi^\# i^\# \\
 &= \ni_A, & \{ \text{Prop. 12(e)} : i \xi i^\# = \text{id}_A \}
 \end{aligned}$$

which shows $\text{sup}(\ni_X i, \xi) t = \text{id}_{\wp(A)}$ by the extensionality of membership relations. Therefore t is a surjection, for $t^\# t = \text{sup}(\ni_X i, \xi) t t^\# t = \text{sup}(\ni_X i, \xi) t = \text{id}_{\wp(A)}$. $\square$

9 Conclusions

In this paper we re-formulated the well-known notions of Boolean algebras, the filters and atoms in Dedekind and Schröder categories, and proved their elementary properties. In particular, we have seen that a power object in Schröder categories forms a complete Boolean algebra, and that the prime, ultra and maximal filters are evidently equivalent, and then extended the representability theorems of Boolean algebras by atoms and Stone maps in the relational categories. As the proofs of Prop. 7 (Cf. Prop. 10) and Theorem 2 still depend on Zorn's lemma in set theory and on the strict point axiom, respectively, the demonstration of these theorems completely within the relational categories is one of our future works. Last but not least Schmidt [11] studied the concept of Boolean lattices as well as a lot of common relational notions stated in the present paper. It is obviously interesting to investigate a relationship between Schmidt's Boolean lattices and ours.

Acknowledgements. The author is grateful to anonymous referees for helpful comments and suggestions.

References

1. Desharnais, J.: Monomorphic characterization of n -ary direct products. *Information Sciences* 119(3-4), 275–288 (1999)
2. Duntsch, I., Winter, M.: A representation theorem for Boolean contact algebras. *Theoretical Computer Science* 347(3), 498–512 (2005)
3. Freyd, P., Scedrov, A.: *Categories, allegories*. North-Holland, Amsterdam (1990)
4. Ishida, T., Honda, K., Kawahara, Y.: Formal concepts in Dedekind categories (to appear in this volume)
5. Kawahara, Y.: Lattices in Dedekind categories. In: Orłowska, E., Szalas, A. (eds.) *Relational Methods for Computer Science Applications*, pp. 247–260. Physica-Verlag (2001)
6. Kawahara, Y.: Groups in allegories. In: de Swart, H. (ed.) *RelMiCS 2001*. LNCS, vol. 2561, pp. 88–103. Springer, Heidelberg (2002)
7. Kawahara, Y.: Urysohn's lemma in Schröder categories. *Bull. Inform. Cybernet* 39, 69–81 (2007)
8. Mac Lane, S.: *Categories for the working mathematician*. Springer, Heidelberg (1999)
9. Olivier, J.-P., Serrato, D., de Dedekind, C.: Morphismes dans les Catégories de Schröder. *C. R. Acad. Sci. Paris* 260, 939–941 (1980)
10. Schmidt, G., Ströhlein, T.: *Relations and graphs. Discrete Mathematics for Computer Science*. Springer, Berlin (1993)
11. Schmidt, G.: Partiality I: Embedding relation algebras. *JLAP* 66, 212–238 (2006)
12. Schmidt, R.A. (ed.): *RelMiCS/AKA 2006*. LNCS, vol. 4136. Springer, Heidelberg (2006)
13. Tarski, A.: On the calculus of relations. *J. Symbolic Logic* 6, 73–89 (1941)

Cardinality in Allegories

Yasuo Kawahara¹ and Michael Winter^{2,*}

¹ Department of Informatics,
Kyushu University,
Fukuoka, Japan

`kawahara@i.kyushu-u.ac.jp`

² Department of Computer Science,
Brock University,
St. Catharines, Ontario, Canada, L2S 3A1
`mwinter@brocku.ca`

Abstract. In this paper we want to investigate two notions of the cardinality of relations in the context of allegories. The different axiom systems are motivated on the existence of injective and surjective functions, respectively. In both cases we provide a canonical cardinality function and show that it is initial in the category of all cardinality functions over the given allegory.

1 Introduction

The calculus of relations, and its categorical versions in particular, are often used to model programming languages, classical and non-classical logics and different methods of data mining (see for example [12, 3, 8, 9]). In many applications relations that are minimal with respect to inclusion as well as minimal with respect to their cardinality are substantial. For example, in deductive databases and logic programming the minimal relations satisfying all facts and rules taken as the semantics of the program. Another example arises from non-monotonic reasoning, where minimality is crucial in formalizing abnormal behaviors and situations. The last example is taken from graph theory. Finite trees can be characterized as those connected graphs satisfying the numerical equation $e = n - 1$ relating the number of edges e and vertices n . Since graphs can be considered as binary relation, an abstract formulation of the property above in the theory of allegories needs a notion of cardinality.

In this paper we want to investigate two notions of the cardinality of relations in the context of allegories. The first notion is motivated by the standard cardinal (pre)ordering of sets, i.e. a set A is smaller than a set B if there is an injective function from A to B . The second notion will be based on surjective functions, i.e. we consider a set A smaller than a set B if there is a surjective function from B to A . Ignoring the empty set, the two notions are equivalent in regular set

* The author gratefully acknowledges support from the Natural Sciences and Engineering Research Council of Canada.

theory with the axiom of choice. Since the theory of allegories is much weaker we cannot expect such a result in general.

In both cases we provide a canonical cardinality function and show that it is initial in the category of all cardinality functions over the given allegory. Last but not least, we give an additional axiom characterizing the canonical cardinality function (up to isomorphism).

2 Categories of Relations

Throughout this paper we assume that the reader is familiar with the basic notions from category and lattice theory. For notions not defined here we refer to [4, 5].

Given a category $\mathcal{C}$ we denote its collection of objects by $\text{Obj}_{\mathcal{C}}$ and its collection of morphisms by $\text{Mor}_{\mathcal{C}}$. To indicate that a morphism f has source A and target B we usually write $f : A \rightarrow B$. The collection of all morphisms between A and B is denoted by $\mathcal{C}[A, B]$. We use $;$ for composition of morphisms, which has to be read from left to right, i.e. $f;g$ means first f then g . The identity morphism on the object A is written as $\mathbb{I}_A$.

Definition 1. *An allegory $\mathcal{R}$ is a category satisfying the following:*

1. *For all objects A and B the class $\mathcal{R}[A, B]$ is a lower semi-lattice. Meet and the induced ordering are denoted by $\sqcap, \sqsubseteq$, respectively. The elements in $\mathcal{R}[A, B]$ are called relations.*
2. *There is a monotone operation $\smile$ (called the converse operation) such that for all relations $Q, R : A \rightarrow B$ and $S : B \rightarrow C$ the following holds*

$$(Q;S)^{\smile} = S^{\smile};Q^{\smile} \quad \text{and} \quad (Q^{\smile})^{\smile} = Q.$$

3. *For all relations $Q : A \rightarrow B, R, S : B \rightarrow C$ we have $Q;(R \sqcap S) \sqsubseteq Q;R \sqcap Q;S$.*
4. *For all relations $Q : A \rightarrow B, R : B \rightarrow C$ and $S : A \rightarrow C$ the following modular law holds $Q;R \sqcap S \sqsubseteq Q;(R \sqcap Q^{\smile};S)$.*

A relation $R : A \rightarrow B$ is called univalent (or a partial function) iff $R^{\smile};R \sqsubseteq \mathbb{I}_B$ and total iff $\mathbb{I}_A \sqsubseteq R;R^{\smile}$. Functions are total and univalent relations and are usually denoted by lowercase letters. Furthermore, R is called injective iff $R^{\smile}$ is univalent and surjective iff $R^{\smile}$ is total. In the following lemma we have summarized several basic properties of relations used in this paper. A proof can be found in [4, 8, 9].

Lemma 1. *Let $\mathcal{R}$ be an allegory. Then we have:*

1. $Q;R \sqcap S \sqsubseteq (Q \sqcap S;R^{\smile});(R \sqcap Q^{\smile};S)$ for all relations $Q : A \rightarrow B, R : B \rightarrow C$ and $S : A \rightarrow C$ (Dedekind formula);
2. If $Q : A \rightarrow B$ is univalent, then $Q;(R \sqcap S) = Q;R \sqcap Q;S$ for all relations $R, S : B \rightarrow C$;
3. If $R : B \rightarrow C$ is univalent, then $Q;R \sqcap S = (Q \sqcap S;R^{\smile});R$ for all relations $Q : A \rightarrow B$ and $S : A \rightarrow C$.

Another important property of commuting squares of functions is as follows:

Lemma 2. *Let $\mathcal{R}$ be an allegory, and $f : A \rightarrow B, g : A \rightarrow C, h : B \rightarrow D$ and $k : C \rightarrow D$ be functions with $f^\sim; g = h; k^\sim$. Then we have $f; h = g; k$.*

Proof. Consider the following computation

$$\begin{array}{ll}
 f; h \sqsubseteq g; g^\sim; f; h & g \text{ total} \\
 = g; k; h^\sim; h & \text{assumption} \\
 \sqsubseteq g; k & h \text{ univalent} \\
 \sqsubseteq f; f^\sim; g; k & f \text{ total} \\
 = f; h; k^\sim; k & \text{assumption} \\
 \sqsubseteq f; h. & k \text{ univalent}
 \end{array}$$

This completes the proof. $\square$

Two functions $f : C \rightarrow A$ and $g : C \rightarrow B$ with common source are said to tabulate a relation $R : A \rightarrow B$ iff $R = f^\sim; g$ and $f; f^\sim \sqcap g; g^\sim = \mathbb{I}_C$. If for all relations of an allegory $\mathcal{R}$ there is tabulation, then $\mathcal{R}$ is called tabular. Notice that a function $f : A \rightarrow B$ and its converse $f^\sim : B \rightarrow A$ always have a tabulation. The tabulation is given by $(\mathbb{I}_A, f)$ and $(f, \mathbb{I}_B)$, respectively.

Lemma 3. *Let $\mathcal{R}$ be an allegory, and $R : A \rightarrow B$ a relation that is tabulated by $f : C \rightarrow A$ and $g : C \rightarrow B$. Furthermore, let $h : D \rightarrow A$ and $k : D \rightarrow B$ be functions with $h^\sim; k \sqsubseteq R$, and define $l := h; f^\sim \sqcap k; g^\sim : D \rightarrow C$. Then we have the following:*

1. l is the unique function with $h = l; f$ and $k = l; g$.
2. If $h^\sim; k = R$, then l is surjective.
3. If $h : D \rightarrow A$ and $k : D \rightarrow B$ is a tabulation, i.e. $h; h^\sim \sqcap k; k^\sim = \mathbb{I}_D$, then l is injective.
4. If R is a partial identity, i.e. $A = B$ and $R \sqsubseteq \mathbb{I}_A$, then f (or g) is a tabulation of R , i.e. $R = f^\sim; f$ and $f; f^\sim = \mathbb{I}_C$.

Proof. 1. This was already shown in 2.143 of [4].

2. Assume $h^\sim; k = R$. Then we have

$$\begin{array}{ll}
 \mathbb{I}_C = \mathbb{I}_C \sqcap f; f^\sim; g; g^\sim & f, g \text{ total} \\
 = \mathbb{I}_C \sqcap f; h^\sim; k; g^\sim & \text{assumption} \\
 \sqsubseteq (f; h^\sim \sqcap g; k^\sim); (h; f^\sim \sqcap k; g^\sim) & \text{Lemma 1(1)} \\
 = l^\sim; l.
 \end{array}$$

3. Assume $h; h^\sim \sqcap k; k^\sim = \mathbb{I}_D$. Then we have

$$\begin{array}{ll}
 l; l^\sim = (h; f^\sim \sqcap k; g^\sim); (f; h^\sim \sqcap g; k^\sim) & \\
 \sqsubseteq h; f^\sim; f; h^\sim \sqcap k; g^\sim; g; k^\sim & \\
 \sqsubseteq h; h^\sim \sqcap k; k^\sim & f, g \text{ univalent} \\
 = \mathbb{I}_D. & \text{assumption}
 \end{array}$$

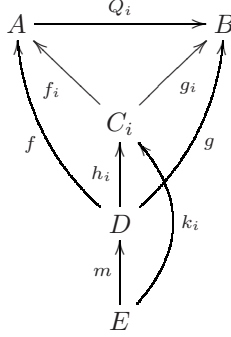
4. This was already shown in 2.145 of [4]. $\square$

The previous lemma also implies that tabulations are unique up to isomorphism.

The next lemma is concerned with a tabulation of the meet of two relations.

Lemma 4. *Let $\mathcal{R}$ be an allegory, and $Q_i : A \rightarrow B$ be relations tabulated by $f_i : C_i \rightarrow A$ and $g_i : C_i \rightarrow B$ for $i = 1, 2$. If $f : D \rightarrow A$ and $g : D \rightarrow B$ is a tabulation of $Q_1 \sqcap Q_2$, then there are unique injections $h_i : D \rightarrow C_i$ ($i = 1, 2$) satisfying the following:*

1. $h_i; f_i = f$ and $h_i; g_i = g$;
2. If there are functions $k_i : E \rightarrow C$ with $k_1; f_1 = k_2; f_2$ and $k_1; g_1 = k_2; g_2$, then there is a unique function $m : E \rightarrow D$ with $k_i = m; h_i$ ($i = 1, 2$).



Proof. From Lemma 3 (1) and (3) we get $h_i = f; f_i^\sim \sqcap g_i; g_i^\sim$. It just remains to verify the second property. Assume $k_i : E \rightarrow C$ are as required, and let $p := k_1; f_1 = k_2; f_2$ and $q := k_1; g_1 = k_2; g_2$. Then we have

$$\begin{aligned}
 p^\sim; q &= p^\sim; q \sqcap p^\sim; q \\
 &= (k_1; f_1)^\sim; k_1; g_1 \sqcap (k_2; f_2)^\sim; k_2; g_2 && \text{by definition} \\
 &= f_1^\sim; k_1^\sim; k_1; g_1 \sqcap f_2^\sim; k_2^\sim; k_2; g_2 \\
 &\sqsubseteq f_1^\sim; g_1 \sqcap f_1^\sim; g_1 && k_i \text{ univalent} \\
 &= Q_1 \sqcap Q_2.
 \end{aligned}$$

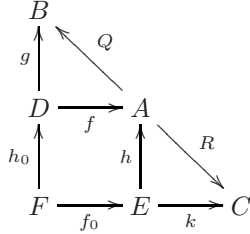
Since f, g is a tabulation of $Q_1 \sqcap Q_2$ there is a unique function $m : E \rightarrow D$ with $m; f = p$ and $m; g = q$. We conclude $m; h_i; f_i = m; f = p = k_i; f_i$ and $m; h_i; g_i = m; g = q = k_i; g_i$ for $i = 1, 2$. This implies

$$\begin{aligned}
 m; h_i &= m; h_i; (f_i; f_i^\sim \sqcap g_i; g_i^\sim) && f_i, g_i \text{ is a tabulation} \\
 &= m; h_i; f_i; f_i^\sim \sqcap m; h_i; g_i; g_i^\sim && \text{Lemma 1(2)} \\
 &= k_i; f_i; f_i^\sim \sqcap k_i; g_i; g_i^\sim && \text{see above} \\
 &= k_i; (f_i; f_i^\sim \sqcap g_i; g_i^\sim) && \text{Lemma 1(2)} \\
 &= k_i. && f_i, g_i \text{ is a tabulation}
 \end{aligned}$$

Suppose $n : E \rightarrow D$ is another function with $n; h_i = k_i$. Then $n; f = n; h_i; f_i = k_i; f_i = p$ and $n; g = n; h_i; g_i = k_i; g_i = q$ so that we conclude $n = m$. $\square$

The last lemma of this section is a technical lemma that will be used in Section 5.

Lemma 5. Let $\mathcal{R}$ be an allegory, and $Q : A \rightarrow B$ and $R : A \rightarrow C$ be relations tabulated by $f : D \rightarrow A$, $g : D \rightarrow B$ and $h : E \rightarrow A$, $k : E \rightarrow C$, respectively. Furthermore, let $h_0 : F \rightarrow D$, $f_0 : F \rightarrow E$ be a tabulation of f ; $h^\sim$. Then $Q; Q^\sim \sqcap R; R^\sim \sqsubseteq \mathbb{I}_A$ iff $h_0; g; g^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim = \mathbb{I}_F$.



Proof. ' $\Rightarrow$ ': Assume $Q; Q^\sim \sqcap R; R^\sim \sqsubseteq \mathbb{I}_A$. Then we have

$$\begin{aligned}
 & h_0; g; g^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim \\
 & \sqsubseteq h_0; f; f^\sim; g; g^\sim; f; f^\sim; h_0^\sim \sqcap f_0; h; h^\sim; k; k^\sim; h; h^\sim; f_0^\sim && f, h \text{ total} \\
 & = f_0; h; f^\sim; g; g^\sim; f; h^\sim; f_0^\sim \sqcap f_0; h; h^\sim; k; k^\sim; h; h^\sim; f_0^\sim && \text{Lemma 2} \\
 & = f_0; h; (f^\sim; g; g^\sim; f \sqcap h^\sim; k; k^\sim; h); h^\sim; f_0^\sim && \text{Lemma 1(2)} \\
 & = f_0; h; (Q; Q^\sim \sqcap R; R^\sim); h^\sim; f_0^\sim && \text{tabulations} \\
 & \sqsubseteq f_0; h; h^\sim; f_0^\sim. && \text{assumption}
 \end{aligned}$$

We conclude

$$\begin{aligned}
 & h_0; g; g^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim \\
 & = h_0; g; g^\sim; h_0^\sim \sqcap f_0; h; h^\sim; f_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim \sqcap f_0; h; h^\sim; f_0^\sim && \text{see above} \\
 & = h_0; g; g^\sim; h_0^\sim \sqcap h_0; f; f^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim \sqcap f_0; h; h^\sim; f_0^\sim && \text{Lemma 2} \\
 & = h_0; (g; g^\sim \sqcap f; f^\sim); h_0^\sim \sqcap f_0; (k; k^\sim \sqcap h; h^\sim); f_0^\sim && \text{Lemma 1(2)} \\
 & = h_0; h_0^\sim \sqcap f_0; f_0^\sim && \text{tabulations} \\
 & = \mathbb{I}_F. && \text{tabulation}
 \end{aligned}$$

' $\Leftarrow$ ': Now, assume $h_0; g; g^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim = \mathbb{I}_F$. Then we have

$$\begin{aligned}
 & Q; Q^\sim \sqcap R; R^\sim \\
 & = f^\sim; g; g^\sim; f \sqcap h^\sim; k; k^\sim; h && \text{tabulation} \\
 & = f^\sim; (g; g^\sim; f; h^\sim \sqcap f; h^\sim; k; k^\sim); h && \text{Lemma 1(3)} \\
 & = f^\sim; (g; g^\sim; h_0^\sim; f_0 \sqcap h_0^\sim; f_0; k; k^\sim); h && \text{tabulation} \\
 & = f^\sim; h_0^\sim; (h_0; g; g^\sim; h_0^\sim \sqcap f_0; k; k^\sim; f_0^\sim); f_0; h && \text{Lemma 1(3)} \\
 & = f^\sim; h_0^\sim; f_0; h && \text{assumption} \\
 & = f^\sim; f; h^\sim; h && \text{tabulation} \\
 & \sqsubseteq \mathbb{I}_A. && f, h \text{ univalent}
 \end{aligned}$$

This completes the proof. $\square$

Notice that in the situation of the previous lemma we always have

$$g^\sim; h_0^\sim; f_0; k = g^\sim; f; h^\sim; k = Q^\sim; R$$

so that the assertion could be formulated alternatively as follows:

$$Q; Q^\sim \sqcap R; R^\sim \sqsubseteq \mathbb{I}_A \text{ iff } h_0; g \text{ and } f_0; k \text{ is a tabulation of } Q^\sim; R.$$

3 Cardinal Preorderings on Objects

In this section we want to study two notions of preordering on the class of objects of an allegory.

Definition 2. Let $\mathcal{R}$ be an allegory. Then the relations $\lesssim_i$ and $\lesssim_s$ on the class of objects of $\mathcal{R}$ are defined by

1. $A \lesssim_i B$ iff there is an injective function $f : A \rightarrow B$;
2. $A \lesssim_s B$ iff there is a surjective function $f : B \rightarrow A$.

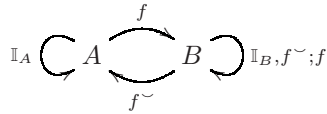
By $\sim_i$ and $\sim_s$ we denote the equivalence relations on the class of objects induced by $\lesssim_i$ and $\lesssim_s$, respectively.

In set theory (with the axiom of choice) both notions are equivalent for non-empty sets. Since the theory of allegories is much weaker we cannot expect the same for arbitrary allegories. We want to give several examples showing that $\lesssim_i$ and $\lesssim_s$ are different in general - even in the case of tabular allegories.

Example 1. Consider the structure consisting of two sets $A := \{1\}$ and $B := \{1, 2\}$ as objects and the following morphisms:

- The identity relations on A and B .
- The inclusion function $f := \{(1, 1)\}$ from A to B and its converse.
- The partial identity $f^\sim; f = \{(1, 1)\}$ on B .

The structure can be visualized by the following graph:



It is easy to verify that this structure is closed under composition, converse and intersection, and is, therefore, an allegory. Furthermore, this allegory is tabular. The only relation that is not a function or a converse of a function is $f^\sim; f$, which is tabulated by the pair (f, f) .

f is an injective function so that we get $A \lesssim_i B$. On the other hand, there is no surjective function from B to A so that $A \lesssim_s B$ does not hold. The order structure induced by $\lesssim_s$ is discrete whereas the order structure induced by $\lesssim_i$ is linear.

This example can be extended by adding the objects $\{1, 2, 3\}, \{1, 2, 3, 4\}, \dots$ and the corresponding inclusion functions. $\lesssim_s$ remains to be discrete and $\lesssim_i$ is linear of length ω .

Example 2. Let $R_n^p \subseteq \omega \times \omega$ with $n \geq 0$ and p an arbitrary integer be defined by

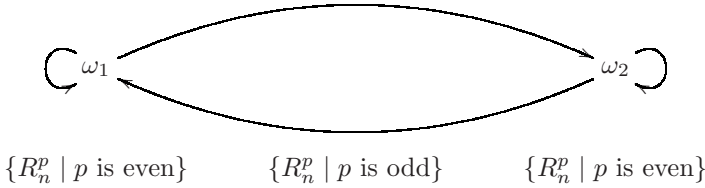
$$(x, y) \in R_n^p : \iff x + p = y \text{ and } \min(x, y) \geq n.$$

It is easy to verify that the following properties are satisfied:

1. $R_0^0 = \mathbb{I}_\omega$,
2. $(R_n^p)^\smile = R_n^{-p}$
3. $R_m^p \sqcap R_n^q = \begin{cases} \emptyset & : p \neq q \\ R_{\max(m,n)}^p & : p = q, \end{cases}$
4. $R_m^p ; R_n^q = R_{\max(m, m-p, n, n+q)}^{p+q}$ for an $l \geq 0$.

The properties above show that the set of relations $\{R_n^p \mid n \geq 0, p \in \mathbb{Z}\}$ is closed under all operations of an allegory.

Consider the allegory given by two copies of the natural numbers ω_1, ω_2 and the morphism sets as indicated in the following diagram:



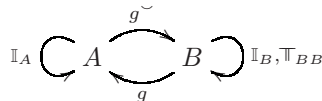
In this allegory there is an injection $R_0^1 : \omega_1 \rightarrow \omega_2$ (the successor function). By the symmetric definition of the allegory the same relation is also an injection from ω_2 to ω_1 . The only bijection R_0^0 is not a relation between ω_1 and ω_2 since its exponent is even. Notice that R_0^0 is also the only surjective function in the given set of relations. Consequently, $\omega_1 \sim_i \omega_2$ but we have neither $\omega_1 \lesssim_s \omega_2$ nor $\omega_2 \lesssim_s \omega_1$.

This example is pre-tabular, i.e. every relation is included in a tabular relation. This follows from the fact that every relation is included in an injection or in the converse of such a relation. The embedding of a pre-tabular in a tabular allegory by splitting partial identities is full. Consequently, the resulting allegory permits the same example as above but is tabular.

Example 3. Again, consider the structure consisting of the two sets $A := \{1\}$ and $B := \{1, 2\}$ as objects and the following morphisms:

- The identity relations on A and B .
- The function $g := \{(1, 1), (2, 1)\}$ from B to A and its converse.
- The universal relation $\mathbb{I}_{BB} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$ on B .

The structure can be visualized by the following graph:



It is easy to verify that this structure is closed under composition, converse and intersection, and is, therefore, an allegory. This allegory is not tabular since $\mathbb{T}_{BB}$ has no tabulation.

g is a surjection so that we get $A \lesssim_s B$, but there is no injective function from A to B so that $A \lesssim_i B$ does not hold.

There is also an example of tabular allegory containing two objects A and B with $A \lesssim_s B$ and $A \not\lesssim_i B$. This example uses a substructure of a model of ZF not satisfying the axiom of choice and its tabular closure within the given model of set theory. Details can be found in [7].

4 Cardinality Function (Injective Case)

We now give the definition of cardinality function motivated by the preordering $\lesssim_i$.

Definition 3. Let $\mathcal{R}$ be an allegory, and $(\mathcal{C}, \leq)$ be a (partially) ordered class. A function $|\cdot|_i : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{C}$ mapping the morphisms of $\mathcal{R}$ to elements of $\mathcal{C}$ is called a (injective) cardinality function iff

C0: $|R^\sim|_i = |R|_i$ for all relations R ;

I1: $|\cdot|_i$ is monotonic, i.e. $R \sqsubseteq S$ implies $|R|_i \leq |S|_i$ for all relations $R, S : A \rightarrow B$;

I2: If $U : C \rightarrow A$ and $V : C \rightarrow B$ are univalent with $U; U^\sim \sqcap V; V^\sim \sqsubseteq \mathbb{I}_C$, then

$$|U^\sim; V|_i = |U; U^\sim \sqcap V; V^\sim|_i.$$

$|\cdot|_i$ is called strong iff it is surjective as a function and $|\mathbb{I}_A|_i \leq |\mathbb{I}_B|_i$ implies that there is an injection $i : A \rightarrow B$.

The first axiom has its obvious motivation in concrete relations. All versions of cardinality functions in this paper use this axiom so that we call it C0. It turns out in the next section that the second axiom actually characterizes the usage of injective functions. An immediate consequence of the last axiom (see Lemma 6(2)) is that one may compute the cardinality of a relation using its tabulation (if it exists). This idea is the motivation of Axiom (3). We will show later that the strong property makes the cardinality function unique (up to isomorphism).

The first part of the next lemma shows that an (injective) cardinality function is based on the preordering $\lesssim_i$.

Lemma 6. Let $|\cdot|_i$ be a cardinality function over the allegory $\mathcal{R}$. Then:

1. If $i : A \rightarrow B$ is an injection, then $|\mathbb{I}_A|_i \leq |\mathbb{I}_B|_i$.
2. If $R : A \rightarrow B$ has a tabulation $f : C \rightarrow A$ and $g : C \rightarrow B$, then $|R|_i = |\mathbb{I}_C|_i$.

Proof. 1. i is univalent and we have $i; i^\sim = i; i^\sim \sqcap i; i^\sim = \mathbb{I}_A$ since i is total and injective so that Axiom I2 shows $|\mathbb{I}_A|_i = |i^\sim; i|_i$. The latter is less than or equal to $|\mathbb{I}_B|_i$, which follows from $i^\sim; i \sqsubseteq \mathbb{I}_B$ by Axiom I1.

2. This is an immediate consequence of Axiom I2 since f and g are functions with $f; f^\sim \sqcap g; g^\sim = \mathbb{I}_C$ and $R = f^\sim; g$. $\square$

In order to define the canonical cardinality function on allegories for the injective case we need tabulations. Consequently, we will assume for the rest of this section that the given allegory $\mathcal{R}$ is tabular.

Let us denote by $[A]_i$ the equivalence class of an object with respect to $\sim_i$ and by $(\text{Obj}_{\mathcal{R}} / \sim_i, \leq_i)$ the ordered class of those equivalence classes.

Definition 4. *The canonical cardinality function $|\cdot|_i^*$ is defined by $|R|_i^* := [C]_i$ where $R : A \rightarrow B$ has a tabulation $f : C \rightarrow A$ and $g : C \rightarrow B$.*

Notice that the canonical cardinality function is well-defined since tabulations are unique up to isomorphism.

Lemma 7. *The canonical cardinality function $|\cdot|_i^*$ is a cardinality function.*

- Proof.* C0: Notice that (g, f) is a tabulation of $R^\sim$ iff (f, g) is a tabulation of R . We conclude $|R|_i^* = [C]_i = |R^\sim|_i^*$.
- I1: Assume $R \sqsubseteq S$, R is tabulated by $f : C \rightarrow A, g : C \rightarrow B$ and S by $h : D \rightarrow A, k : D \rightarrow B$. Then by Lemma 3(3) there is an injection $i : C \rightarrow D$. This implies $|R|_i^* = [C]_i \leq_i [D]_i = |S|_i^*$.
- I2: Assume that $U : C \rightarrow A$ and $V : C \rightarrow B$ are univalent relations with $U; U^\sim \sqcap V; V^\sim \sqsubseteq \mathbb{I}_C$. Since $U; U^\sim \sqcap V; V^\sim$ is a partial identity we conclude from Lemma 3(4) that there is a function $f : D \rightarrow C$ with $U; U^\sim \sqcap V; V^\sim = f^\sim; f$ and $f; f^\sim = \mathbb{I}_D$. The relation $h := f; U$ is univalent because it is the composition of univalent relations. Furthermore, we have

$$\begin{aligned}
 f; U; (f; U)^\sim &= f; U; U^\sim; f^\sim \\
 &\sqsupseteq f; (U; U^\sim \sqcap V; V^\sim); f^\sim \\
 &= f; f^\sim; f; f^\sim && f \text{ tabulates } U; U^\sim \sqcap V; V^\sim \\
 &= \mathbb{I}_D, && \text{see above}
 \end{aligned}$$

i.e. h is a function. Analogously, $k := f; V$ is a function. We get

$$\begin{aligned}
 h^\sim; k &= U^\sim; f^\sim; f; V \\
 &= U^\sim; (U; U^\sim \sqcap V; V^\sim); V && f \text{ tabulates } U; U^\sim \sqcap V; V^\sim \\
 &= U^\sim; V. && \text{Lemma 1(3)}
 \end{aligned}$$

We conclude that $h : D \rightarrow A, k : D \rightarrow B$ is a tabulation of $U^\sim; V$, and, hence, $|U^\sim; V|_i^* = [D]_i = |U; U^\sim \sqcap V; V^\sim|_i^*$. $\square$

In order to characterize the canonical cardinality function we use the category $\text{Card}_i(\mathcal{R})$. The objects of this category are the cardinality functions based on

$\mathcal{R}$. A morphism between two cardinality functions $|\cdot|_i^1 : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{C}_1$ and $|\cdot|_i^2 : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{C}_2$ is a monotonic function $G : \mathcal{C}_1 \rightarrow \mathcal{C}_2$ so that the following diagram commutes:

$$\begin{array}{ccc} & \mathcal{R} & \\ |\cdot|_i^1 \swarrow & & \searrow |\cdot|_i^2 \\ \mathcal{C}_1 & \xrightarrow{G} & \mathcal{C}_2 \end{array}$$

Theorem 1. *A strong cardinality function is an initial object of $\text{Card}_i(\mathcal{R})$.*

Proof. Assume $|\cdot|_i^s : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{D}$ is a strong cardinality function. First, we want to show that every element of $\mathcal{D}$ is image of an identity relation via $|\cdot|_i^s$. Let x be an element of $\mathcal{D}$. Since $|\cdot|_i^s$ is strong there is a relation $R : A \rightarrow B$ with $|R|_i^s = x$. Let $f : C \rightarrow A$ and $g : C \rightarrow B$ be a tabulation of R . Then by Lemma 6(2) we have $|\mathbb{I}_C|_i^s = |R|_i^s = x$.

Let $|\cdot|_i : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{C}$ be an arbitrary cardinality function, and define $G(x) := |\mathbb{I}_A|_i$ with $|\mathbb{I}_A|_i^s = x$. We have to show that G is well-defined, i.e. it is independent of the choice of $\mathbb{I}_A$. Assume $|\mathbb{I}_A|_i^s = |\mathbb{I}_B|_i^s = x$. Since $|\cdot|_i^s$ is strong there are injections $i_1 : A \rightarrow B$ and $i_2 : B \rightarrow A$. By Lemma 6(2) we conclude $|\mathbb{I}_A|_i = |\mathbb{I}_B|_i$. A similar argument shows that G is also monotonic.

Now, let $R : A \rightarrow B$ be a relation and $f : C \rightarrow A$ and $g : C \rightarrow B$ a tabulation of R . Then we have $G(|R|_i^s) = |\mathbb{I}_C|_i = |R|_i$ again by Lemma 6(2). G is obviously the unique function with that property. $\square$

The canonical cardinality function is strong by definition so that we get the following corollary:

Corollary 1. *The canonical cardinality function is an initial object of $\text{Card}_i(\mathcal{R})$.*

A further consequence is that any initial object of $\text{Card}_i(\mathcal{R})$ must be strong because it is isomorphic to the canonical cardinality function.

Corollary 2. *A cardinality function is an initial object of $\text{Card}_i(\mathcal{R})$ iff it is strong.*

5 Cardinality Function (Surjective Case)

We now give the definition of cardinality function motivated by the preordering $\lesssim_s$.

Definition 5. *Let $\mathcal{R}$ be an allegory, and $(\mathcal{C}, \leq)$ be an ordered class. A function $|\cdot|_s : \text{Mor}_{\mathcal{R}} \rightarrow \mathcal{C}$ mapping the morphisms of $\mathcal{R}$ to elements of $\mathcal{C}$ is called a (surjective) cardinality function iff*

C0: $|R^\sim|_s = |R|_s$ for all relations R ;

S1: If $Q; Q^\sim \sqcap S; S^\sim \sqsubseteq \mathbb{I}_B$ for relations $Q : A \rightarrow B$ and $S : A \rightarrow C$, then for all $R : B \rightarrow C$

$$|Q; R \sqcap S|_s \leq |R \sqcap Q^\sim; S|_s.$$

$|\cdot|_s$ is called strong iff it is surjective as a function and $|\mathbb{I}_A|_s \leq |\mathbb{I}_B|_s$ implies that there is a surjection $s : B \rightarrow A$.

S1 is also called the Dedekind inequality because of its similarity to the Dedekind formula. Notice that a weaker version was already used in [6].

The first part of the next lemma shows that a (surjective) cardinality function is based on the preordering $\lesssim_s$.

Lemma 8. *Let $|\cdot|_s$ be a cardinality function over the allegory $\mathcal{R}$. Then:*

1. *If $s : B \rightarrow A$ is a surjection, then $|\mathbb{I}_A|_s \leq |\mathbb{I}_B|_s$.*
2. *Axiom I2 is valid.*
3. *If $R : A \rightarrow B$ has a tabulation $f : C \rightarrow A$ and $g : C \rightarrow B$, then $|R|_s = |\mathbb{I}_C|_s$.*

Proof. 1. We have $s^\smile; s = \mathbb{I}_A$ and $\mathbb{I}_B \subseteq s; s^\smile$ and conclude

$$\begin{aligned}
 |\mathbb{I}_A|_s &= |\mathbb{I}_A \cap s^\smile; s|_s & s^\smile; s &= \mathbb{I}_A \\
 &\leq |s \cap s|_s & \text{S1 since } s^\smile; s \cap \mathbb{I}_A; \mathbb{I}_A^\smile &\subseteq \mathbb{I}_A \\
 &= |s^\smile \cap s^\smile|_s & \text{C0} \\
 &\leq |s; s^\smile \cap \mathbb{I}_B|_s & \text{S1 since } s^\smile; s &\subseteq \mathbb{I}_A \\
 &= |\mathbb{I}_B|_s.
 \end{aligned}$$

2. Let $U : C \rightarrow A$ and $V : C \rightarrow B$ be univalent relations with $U; U^\smile \cap V; V^\smile \subseteq \mathbb{I}_C$. Then the assertion follows from

$$\begin{aligned}
 |U^\smile; V|_s &= |U^\smile; V \cap U^\smile; V|_s \\
 &\leq |V \cap U; U^\smile; V|_s & \text{S1 since } U^\smile; U &\subseteq \mathbb{I}_A \\
 &= |V^\smile; U; U^\smile \cap V^\smile|_s & \text{C0} \\
 &\leq |U; U^\smile \cap V; V^\smile|_s & \text{S1 since } V^\smile; V &\subseteq \mathbb{I}_B \\
 &\leq |U^\smile \cap U^\smile; V; V^\smile|_s & \text{S1 since } U; U^\smile \cap V; V^\smile; V; V^\smile & \\
 & & & \subseteq U; U^\smile \cap V; V^\smile \subseteq \mathbb{I}_C \\
 &= |V; V^\smile; U \cap U|_s & \text{C0} \\
 &\leq |V^\smile; U \cap V^\smile; U|_s & \text{S1 since } V; V^\smile \cap U; U^\smile &\subseteq \mathbb{I}_C \\
 &= |U^\smile; V|_s. & \text{C0}
 \end{aligned}$$

3. This property uses the same proof as in Lemma 6(2) using (2) of the current lemma. Notice that monotonicity of the cardinality function is not used in that proof. $\square$

Again, we are just able to define the canonical cardinality function using tabulations. Therefore, we will assume for the rest of this section that the given allegory $\mathcal{R}$ is tabular.

As before, let us denote by $[A]_s$ the equivalence class of an object with respect to $\sim_i s$ and by $(\text{Obj}_{\mathcal{R}} / \sim_s, \leq_s)$ the ordered class of those equivalence classes.

Definition 6. The canonical cardinality function $|\cdot|_s^*$ is defined by $|R|_s^* := [C]_i$ where $R : A \rightarrow B$ has a tabulation $f : C \rightarrow A$ and $g : C \rightarrow B$.

Notice that the canonical cardinality function in the surjective case has the same definition as in the injective case. The main difference is in the ordered classes $(\text{Obj}_{\mathcal{R}} / \sim_i, \leq_i)$ and $(\text{Obj}_{\mathcal{R}} / \sim_s, \leq_s)$.

Lemma 9. The canonical cardinality function $|\cdot|_s^*$ is a cardinality function.

Proof. C0: Analogously to the injective case.

S1: Let $Q : A \rightarrow B$, $R : B \rightarrow C$ and $S : A \rightarrow C$ be relations with $Q; Q^\sim \sqcap S; S^\sim \sqsubseteq \mathbb{I}_B$. Furthermore, suppose that we have the following tabulations:

$$\begin{array}{ll} Q = f_Q^\sim; g_Q, & f_Q; f_Q^\sim \sqcap g_Q; g_Q^\sim = \mathbb{I}_X, \\ R = f_R^\sim; g_R, & f_R; f_R^\sim \sqcap g_R; g_R^\sim = \mathbb{I}_Y, \\ S = f_S^\sim; g_S, & f_S; f_S^\sim \sqcap g_S; g_S^\sim = \mathbb{I}_Z, \\ Q; R = f_{Q;R}^\sim; g_{Q;R}, & f_{Q;R}; f_{Q;R}^\sim \sqcap g_{Q;R}; g_{Q;R}^\sim = \mathbb{I}_U, \\ f_S; f_Q^\sim = h^\sim; k, & h; h^\sim \sqcap k; k^\sim = \mathbb{I}_V, \\ g_Q; f_R^\sim = m^\sim; n, & m; m^\sim \sqcap n; n^\sim = \mathbb{I}_W. \end{array}$$

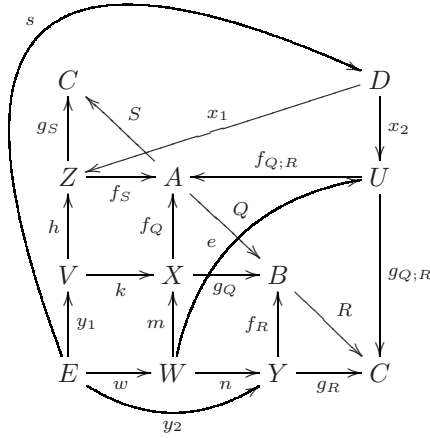
By definition of the canonical cardinality function we get $|Q|_s^* = [X]_s$, $|R|_s^* = [Y]_s$, $|S|_s^* = [Z]_s$ and $|Q; R|_s^* = [U]_s$. Since $Q; Q^\sim \sqcap S; S^\sim \sqsubseteq \mathbb{I}_B$ Lemma 5 shows that $h; g_S$ and $k; g_Q$ is a tabulation of $S^\sim; Q$ so that $|Q^\sim; S|_s^* = [V]_s$ follows. Assume D is the object used in the tabulation of $Q; R \sqcap S$, i.e. $|Q; R \sqcap S|_s^* = [D]_s$. By using the construction of Lemma 4 we obtain injections $x_1 : D \rightarrow Z$ and $x_2 : D \rightarrow U$ with $x_1; f_S = x_2; f_{Q;R}$, $x_1; g_S = x_2; g_{Q;R}$ and $(x_1; f_S)^\sim; x_1; g_S = (x_2; f_{Q;R})^\sim; x_2; g_{Q;R} = Q; R \sqcap S$. Analogously, assuming that $|S; Q^\sim \sqcap R|_s^* = [E]_s$ we obtain two injection $y_1 : E \rightarrow V$ and $y_2 : E \rightarrow Y$ with $y_1; k; g_Q = y_2; f_R$, $y_1; h; g_S = y_2; g_R$ and $(y_1; k; g_Q)^\sim; y_1; h; g_S = (y_2; f_R)^\sim; y_2; g_R = S; Q^\sim \sqcap R$. The following computation

$$\begin{array}{ll} k^\sim; y_1^\sim; y_2 \sqsubseteq k^\sim; y_1^\sim; y_2; f_R; f_R^\sim & f_R \text{ total} \\ = k^\sim; y_1^\sim; y_1; k; g_Q; f_R^\sim & y_1; k; g_Q = y_2; f_R \\ \sqsubseteq g_Q; f_R^\sim & y_1, k \text{ univalent} \end{array}$$

shows that $k^\sim; y_1^\sim; y_2$ is included in the tabulation m, n so that there is a unique function $w : E \rightarrow W$ with $w; m = y_1; k$ and $w; n = y_2$ by Lemma 3(1). Furthermore, we have $Q; R = f_Q^\sim; g_Q; f_R^\sim; g_R = f_Q^\sim; m^\sim; n; g_R$ so that there is a surjection $e : W \rightarrow U$ with $e; f_{Q;R} = m; f_Q$ and $e; g_{Q;R} = n; g_R$ by Lemma 3(2). Finally, consider the computations

$$\begin{array}{ll}
y_1; h; f_S = y_1; k; f_Q & \text{Lemma 2 since } h, k \text{ tabulates } f_S; f_Q^\sim \\
= w; m; f_Q & w; m = y_1; k \\
= w; e; f_{Q;R}, & e; f_{Q;R} = m; f_Q \\
y_1; h; g_S = y_2; g_R & y_1; h; g_S = y_2; g_R \\
= w; n; g_R & w; n = y_2 \\
= w; e; g_{Q;R}. & e; g_{Q;R} = n; g_R
\end{array}$$

From Lemma 4(2) we conclude that there is a unique $s : E \rightarrow D$ with $y_1; h = s; x_1$ and $w; e = s; x_2$. The whole situation is visualized in the following diagram:



It remains to show that s is surjective. First, we have

$$\begin{array}{ll}
s = s; x_1; x_1^\sim & x_1 \text{ injective} \\
= y_1; h; x_1^\sim & y_1; h = s; x_1 \\
= y_1; h; (f_S; f_S^\sim \sqcap g_S; g_S^\sim); x_1^\sim & f_S, g_S \text{ tabulation} \\
= y_1; h; f_S; (x_1; f_S)^\sim \sqcap y_1; h; g_S; (x_1; g_S)^\sim. & \text{Lemma 1(2)}
\end{array}$$

From the computation

$$\begin{array}{ll}
h; f_S = h; f_S \sqcap k; f_Q & \text{Lemma 2} \\
\sqsubseteq h; g_S; g_S^\sim; f_S \sqcap k; g_Q; g_Q^\sim; f_Q & g_S, g_Q \text{ total} \\
= h; g_S; S^\sim \sqcap k; g_Q; Q^\sim, & \\
= h; g_S; g_S^\sim; f_S \sqcap k; g_Q; g_Q^\sim; f_Q & \\
\sqsubseteq (h; g_S; g_S^\sim \sqcap k; g_Q; g_Q^\sim; f_Q; f_S^\sim); f_S & \\
= (h; g_S; g_S^\sim \sqcap k; g_Q; g_Q^\sim; k^\sim; h); f_S & h, k \text{ tabulates } f_S; f_Q^\sim \\
\sqsubseteq (h; g_S; g_S^\sim; h^\sim \sqcap k; g_Q; g_Q^\sim; k^\sim); h; f_S & \\
= h; f_S & \text{Lemma 5}
\end{array}$$

we conclude $h; f_S = h; g_S; S^\sim \sqcap k; g_Q; Q^\sim$. In addition, from

$$\begin{aligned}
& (y_1; h; f_S)^\sim; y_1; h; g_S \\
&= (s; x_1; f_S)^\sim; s; x_1; g_S && y_1; h = s; x_1 \\
&= (x_1; f_S)^\sim; s^\sim; s; x_1; g_S \\
&= (x_1; f_S)^\sim; x_1; g_S && s \text{ univalent} \\
&= Q; R \sqcap S && \text{tabulation} \\
&= Q; R \sqcap S \sqcap S \\
&\sqsubseteq Q; (R \sqcap Q^\sim; S) \sqcap S \\
&= Q; (y_1; k; g_Q)^\sim; y_1; h; g_S \sqcap S && \text{tabulation} \\
&= (k; g_Q; Q^\sim)^\sim; y_1^\sim; y_1; h; g_S \sqcap S \\
&= ((k; g_Q; Q^\sim)^\sim \sqcap S; (y_1^\sim; y_1; h; g_S)^\sim); y_1^\sim; y_1; h; g_S \\
&\sqsubseteq ((k; g_Q; Q^\sim)^\sim \sqcap S; (h; g_S)^\sim); y_1^\sim; y_1; h; g_S && y_1 \text{ univalent} \\
&= (k; g_Q; Q^\sim \sqcap h; g_S; S^\sim)^\sim; y_1^\sim; y_1; h; g_S \\
&= (h; f_S)^\sim; y_1^\sim; y_1; h; g_S && \text{see above} \\
&= (y_1; h; f_S)^\sim; y_1; h; g_S
\end{aligned}$$

we obtain $(y_1; h; f_S)^\sim; y_1; h; g_S = (x_1; f_S)^\sim; x_1; g_S$. Now, we are ready to establish that s is indeed surjective.

$$\begin{aligned}
\mathbb{I}_D &= \mathbb{I}_D \sqcap x_1; f_S; (x_1; f_S)^\sim x_1; g_S; (x_1; g_S)^\sim && x_1, f_S, g_S \text{ total} \\
&= \mathbb{I}_D \sqcap x_1; f_S; (y_1; h; f_S)^\sim; y_1; h; g_S; (x_1; g_S)^\sim && \text{see above} \\
&\sqsubseteq (x_1; f_S; (y_1; h; f_S)^\sim \sqcap x_1; g_S; (y_1; h; g_S)^\sim); && \text{Lemma } \textcolor{red}{1}(1) \\
&\quad (y_1; h; g_S; (x_1; g_S)^\sim \sqcap y_1; h; f_S; (x_1; f_S)^\sim) \\
&= s^\sim; s. && \text{see above}
\end{aligned}$$

This completes the proof. $\square$

As in the injective case we want to characterize the canonical cardinality function. Again we use the category of cardinality functions $\text{Card}_s(\mathcal{R})$, which is defined analogously to $\text{Card}_i(\mathcal{R})$.

Theorem 2. *A strong cardinality function is an initial object of $\text{Card}_s(\mathcal{R})$.*

Proof. The proof of this theorem is similar to the proof of Theorem $\textcolor{red}{1}$ using Lemma $\textcolor{red}{8}(3)$ instead of Lemma $\textcolor{red}{6}(2)$. $\square$

As in the injective case we get the following corollaries:

Corollary 3. *The canonical cardinality function is an initial object of $\text{Card}_s(\mathcal{R})$.*

Corollary 4. *A cardinality function is an initial object of $\text{Card}_s(\mathcal{R})$ iff it is strong.*

6 Conclusion and Outlook

In this paper we have instigated two notions of the cardinality of relations based on the preordering of objects induced by the existence of injective and surjective functions, respectively. An obvious extension is to combine both notion into one concept. The abstract definition will use the Axioms C0, I1 and S1, of course. As the examples in Section 3 show a suitable definition of a canonical cardinality function requires more structure of the underlying allegory. One may require a relational version of the Axiom of Choice:

(AC) For all relations $R : A \rightarrow B$ there is a function $f : A \rightarrow B$ with $f \sqsubseteq R$ and $\mathbb{I}_A \sqcap f; f^\smile = \mathbb{I}_A \sqcap R; R^\smile$.

Notice that the axiom above for tabular power allegories implies that the each lower semi-lattice $\mathcal{R}[A, B]$ is in fact a Boolean algebra. This is just the allegorical version of the fact that the Axiom of Choice in a topos implies that the topos is Boolean.

References

1. Berghammer, R.: Computation of Cut Completions and Concept Lattices Using Relational Algebra and RelView. *JoRMiCS* 1, 50–72 (2004)
2. Bird, R., de Moor, O.: *Algebra of Programming*. Prentice-Hall, Englewood Cliffs (1997)
3. Brink, C., Kahl, W., Schmidt, G. (eds.): *Relational Methods in Computer Science*. Advances in Computer Science Vienna. Springer, Vienna (1997)
4. Freyd, P., Scedrov, A.: *Categories, Allegories*. North-Holland, Amsterdam (1990)
5. Grätzer, G.: *General lattice theory*, 2nd edn. Birkhäuser, Basel (2003)
6. Kawahara, Y.: On the Cardinality of Relations. In: Schmidt, R.A. (ed.) *RelMiCS/AKA 2006*. LNCS, vol. 4136, pp. 251–265. Springer, Heidelberg (2006)
7. Kawahara, Y., Winter, M.: On the Tabular Closure of a Sub-Allegory of a Tabular Allegory (to appear)
8. Schmidt, G., Ströhlein, T.: *Relationen und Graphen*. Springer, Heidelberg (1989); English version: *Relations and Graphs*. Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer, Heidelberg (1993).
9. Winter, M.: *Goguen Categories. A Categorical Approach to L-Fuzzy Relations*. Trends in Logic 25 (2007)

Solving Linear Equations in *-continuous Action Lattices^{*}

Béchir Ktari, François Lajeunesse-Robert, and Claude Bolduc

Département d'informatique et de génie logiciel
Université Laval
Québec, G1K 7P4, Canada

Abstract. This work aims to investigate conditions under which program analysis can be viewed as algebraically solving equations involving terms of subclasses of Kleene algebras and variables. In this paper, we show how to solve a kind of linear equations in which variables appear only on one side of the equality sign, over a *-continuous action lattice. Furthermore, based on the method developed for solving equations, we present how model checking of a restricted version of the linear μ -calculus over finite traces can be done by algebraic manipulations. Finally, we give some ideas on how to extend the resolution method to other classes of equations and algebraic structures.

1 Introduction

Kleene algebras are algebraic structures which are largely used to reason about computer programs. For instance, they could be used to prove the equivalence of two programs [13] by making various algebraic manipulations. However, what happens when two programs are not equivalent? Must we stop there or could it be interesting to go further? One might ask what does it lack to a program so that it would become equivalent to another? This question can be answered by solving equational systems in Kleene algebras. Programs are translated into Kleene algebra expressions and the problem is transformed into resolving equations involving one or several variables. Solutions of these equations indicate how to modify programs so that they become equivalent. Another possible application of the resolution of equations in Kleene algebras comes from model checking. Given a property and a program, it could be useful to know what is missing to a program so that it satisfies the property. As for program equivalence, we can express both the program and the property in Kleene algebra. Then we proceed in a similar way as previously to find possible “refinements” (not necessarily semantics preserving) of the program that satisfy the property. The application to model checking was our initial motivation to investigate the resolution of equations in Kleene algebra.

That being said, solving equations in Kleene algebra, in general, is not an easy task. Depending on the model under which an equation, expressed in the

^{*} This research is supported by a research grant from the Natural Sciences and Engineering Council of Canada, NSERC.

term algebra, is interpreted, it may or may not have a solution. For example, the equation

$$a \cdot X + X \cdot a = 1$$

has a solution in the relational model over the set $\{0, 1\}$ when a is interpreted as the relation $\{(0, 1)\}$ but does not have a solution in the standard language model. So instead of considering the general class of Kleene algebra, we restricted ourselves to a subclass in which we find the most common examples of Kleene algebras [12], namely the $*$ -continuous action lattices. Moreover, for the sake of simplicity, we have decided, from the beginning, to restrict ourselves to the resolution of linear equations. In fact, the higher the degree of an equation is, the higher is the difficulty to solve it.

In universal algebra, unification theory is commonly used to solve equational systems. It consists of finding a substitution which will replace the variables of an equation with different terms of the algebra so that equality holds. For instance, consider the equation

$$pX + tY = Zq + tp$$

where the set of variables is $\{X, Y, Z\}$ and p, t, q are terms of the algebra. In this case, it is easy to see that the substitution $[X/q, Y/p, Z/p]$ is a solution. The concept of unification is general and theoretically applicable to all classes of algebras including Kleene algebras. From this perspective, works were made on the unification of linear equations in semiring [17], which is significantly close to unification in Kleene algebra, an idempotent semiring with axioms defining the Kleene star operator. However, applying the unification theory to a specific algebra can be a tremendous task.

In the literature related to Kleene algebras, we have not found much work related to solving linear equations. The only available work [20] makes use of matrices to solve equations of the form

$$X = aX + b$$

where $X = [X_1 \ X_2 \ \dots \ X_n]^t$, $b = [b_1 \ b_2 \ \dots \ b_n]^t$ and a is a matrix of size $n \times n$. Considering the limitation of this approach, we want to find other techniques for solving a larger class of equations. In particular, we focused on finding some laws and hypotheses allowing us to solve equations in a similar way as we would solve them in classical algebra. Starting from there, we were able to solve linear equations in which the variable appears on one side of the equality sign over idempotent semiring [15]. By restricting ourselves to idempotent semirings, it was possible to identify the conditions under which an equation can be solved.

This paper is an extension of our previous work [15] and it is organized as follows. In Sect. 2, we present the definition of a Kleene algebra in the sense of Kozen [11] and a subalgebra of it, namely $*$ -continuous action lattices, on which equations will be solved. Action lattices discard numbers of Kleene algebras that have undesirable behaviors when it comes to solve equations. The method developed for solving linear equations in which the variable appears only on one side of the equality sign is given in Sect. 3. First, we show that solving

an equation can be reduced to the comparison of two elements of an algebra. Then we present a method for determining if an element is less than or equal to another. Section 4 gives an application of the method developed to verify a program satisfies a property (model-checking). The logic we have considered for this is a restricted version of the linear μ -calculus. Section 5 presents two separate fields of study arisen from the work presented in this paper. Finally, Sect. 6 summarizes the work done and our next objectives.

2 Basics

Idempotent Semirings. An idempotent semiring with identity and neutral element, or idempotent semiring for short, is an algebraic structure $\langle I, +, \cdot, 0, 1 \rangle$ such that, for all $x, y, z \in I$:

$$\begin{array}{ll} x + (y + z) = (x + y) + z & x \cdot 0 = 0 \cdot x = 0 \\ x + 0 = x & x \cdot 1 = x \\ x + y = y + x & 1 \cdot x = x \\ x + x = x & x \cdot (y \cdot z) = (x \cdot y) \cdot z \\ x \cdot (y + z) = x \cdot y + x \cdot z & (x + y) \cdot z = x \cdot z + y \cdot z \end{array}$$

Kleene Algebra. Kleene algebras were developed to answer a question raised by Stephen Cole Kleene asking if it is possible to give a sound and complete axiomatization of the equational theory of regular sets. Since then, different axiomatizations of Kleene algebra have been found. Hereafter, we present the axiomatization proposed by Kozen in [11].

A Kleene algebra is an algebraic structure $\langle K, +, \cdot, *, 0, 1 \rangle$ such that $\langle K, +, \cdot, 0, 1 \rangle$ is an idempotent semiring and that the unary operator $*$ satisfies the axioms:

$$1 + aa^* \leq a^* \quad (1)$$

$$1 + a^*a \leq a^* \quad (2)$$

$$ax \leq x \rightarrow a^*x \leq x \quad (3)$$

$$xa \leq x \rightarrow xa^* \leq x \quad (4)$$

for all $a, x \in K$ and where $\leq$ is the natural partial order over the elements of K , i.e. $x \leq y \leftrightarrow x + y = y$. The precedence between operators, from high to low, is $*, \cdot, +$. We use xy instead of $x \cdot y$ and x^n instead of $\underbrace{x \cdot x \cdot \dots \cdot x}_{n \text{ times}}$.

This class of algebra has been proved to be useful in many applications [13]. Unfortunately, this axiomatization is too permissive when it comes to solve equations. It includes algebras such as the tropical algebra [2] (also named the min, + algebra) which is not a “natural” Kleene algebra. To avoid these unnatural algebras we focus on a subclass of a Kleene algebra.

Residuated Po-Monoid. Residuated po-monoids are algebraic structures that introduce the operators $\backslash$ and $/$, respectively named right and left residual. Intuitively, x/y and $y \backslash x$ can be seen as being a generalization of the division in

classical algebra meaning “ x over y ” and “ y under x ”. In those two cases x corresponds to the dividend while y corresponds to the divisor. Residual structures are an entire field of study independent of Kleene algebras [9]. Here we will refer to them in the context of Kleene algebras in order to have stronger axioms. Therefore, this will discard some of the undesirable algebras.

A residuated po-monoid is an algebraic structure $\langle P, \cdot, 1, \backslash, /, \leq \rangle$ such that $\cdot$ is associative with a two-side identity 1 and such that

$$x \cdot y \leq z \leftrightarrow x \leq z/y \leftrightarrow y \leq x \backslash z \quad (5)$$

for all $x, y, z \in P$, where $\leq$ is a partial order on the elements of P . Besides, the existence of the residuals implies a series of properties. Hereafter, we give some of them that will be useful later (the proofs are given in [16]).

Lemma 1. *Let P be a residuated po-monoid.*

1. *If $\bigsqcup X$ and $\prod Y$ exist for $X, Y \subset P$ then for all $z \in P$, $\prod_{x \in X} x \backslash z$ and $\prod_{y \in Y} z \backslash y$ exist and*

$$\left(\bigsqcup X\right) \backslash z = \prod_{x \in X} x \backslash z \quad \text{and} \quad z \backslash \left(\prod Y\right) = \prod_{y \in Y} z \backslash y$$

2. $1 \backslash x = x$
3. $(xy) \backslash z = y \backslash (x \backslash z)$
4. $x \backslash (y/z) = (x \backslash y)/z$

It should be noted that the previous properties have equivalent mirror forms using the operator $/$ instead of $\backslash$. To obtain them, we have to read the expression backward by substituting $x \cdot y$ by $y \cdot x$ and x/y by $y \backslash x$. Therefore, each time we give a result it is also true for its mirror form.

**-continuous Action Lattice.* As mentioned in [12], action lattices include all common examples of Kleene algebras appearing in automata theory, logics of programs, relational algebra, and the design and analysis of algorithms. This is therefore the most suitable subclass for our desired purpose: the resolution of equations as a technique for program analysis.

An action lattice (see [8]) is an algebraic structure $\langle A, +, \sqcap, \cdot, \backslash, /, *, 0, 1 \rangle$ such that $\langle A, +, \sqcap \rangle$ is a lattice, $\langle A, +, \cdot, *, 0, 1 \rangle$ is a Kleene algebra and such that $\langle A, \cdot, 1, \backslash, / \rangle$ is a residuated po-monoid. The operator $*$ has the higher precedence followed by $\cdot, \backslash$ and $/$ which have the same precedence and by $+$ and $\sqcap$ with the same precedence. It is said to be $*$ -continuous if it also satisfies the axiom

$$ab^*c = \sum_{n \geq 0} ab^n c.$$

Furthermore, since any action lattice is a residuated po-monoid and contains a least element 0, it implies the existence of a greatest element equal to $0/0$, noted ∞ . Moreover, for all $x \in A$ we have $\infty/x = \infty = x/0$.

3 Resolution of Equations in *-continuous Action Lattices

In [15] we have given a definition of linear equations valid for both idempotent semirings and Kleene algebras. Starting from this definition, we identified two classes of linear equations: equations in which variables appear on both sides and equations in which variables appear only on one side of the equality sign. Each one of them requires a different approach for its resolution.

For now, let us consider equations in which the variables appear on one side of the equality sign. These equations have the following form:

$$\sum_{i \in I} a_i X_i b_i + c = d \quad (6)$$

where $a_i, b_i, c, d \in A$, the universe, and X_i are variables for all i belonging to the finite set I . By applying laws of action lattices one can easily find a condition under which this equation has at least one solution. This is given by Corollary 1.

Corollary 1. *A linear equation of the form given in (6) has at least one solution if and only if*

$$c \leq d \quad \bigwedge \quad d \leq \sum_{i \in I} a_i (a_i \backslash d / b_i) b_i + c$$

is valid.

Even if it is easy to characterize whether or not an equation of the form given by (6) has at least one solution it does not mean that we can easily solve such an equation in a *-continuous action lattice. There are basically three main concerns which make it difficult to solve an equation. First, according to Corollary 1, in order to find if an equation has at least a solution we have to check the validity of inequalities. This problem is known to be *PSPACE*-complete [19] for Kleene algebra, in general. While for action lattices it is not known yet if it is decidable or not [12]. Second, there are equations which does not have a least solution. Take for instance the inequality $r \leq X\infty$ interpreted in the language model. Two possible solutions of it are $X = r$ and $X = 1$. But there is no solution to $r \leq X\infty$ which is less than both “ r ” and “ 1 ”. Finally, as we presented earlier there is still equations expressed in the term algebra that have a solution in a certain model of the action lattice and do not have any in an other model.

In response to these concerns we introduce a class of algebra in which it is easy to determine if an inequality is valid. This class of algebra is defined by *-continuous action lattices for which the following hold:

$$y \backslash x = 1 \quad x, y \in \mathcal{G} \text{ and } x = y \quad (7)$$

$$y \backslash x = 0 \quad x \in \mathcal{G} \cup \{0, 1\}, y \in \mathcal{G} \text{ and } x \neq y \quad (8)$$

$$z \backslash (y + x) = z \backslash y + z \backslash x \quad x, y \in A \text{ and } z \in \mathcal{G} \quad (9)$$

$$z \backslash (y \cdot x) = (z \backslash y) \cdot x \quad x \in A \text{ and } y, z \in \mathcal{G} \quad (10)$$

where $\mathcal{G}$ is the finite minimal generative set of the algebra and A is the universe. In the following we will refer to this algebra by $AL_{\mathcal{G}}^*$.

These new axioms are based on theorems of the algebra of regular sets [2,10] over an alphabet Σ , noted Reg_Σ . The algebra of regular sets forms a Kleene algebra according to the standard interpretation $R_\Sigma : T_\Sigma \mapsto \text{Reg}_\Sigma$ defined by

$$R_\Sigma(a) \triangleq \begin{cases} \{a\} & \text{if } a \in \Sigma \\ \{\varepsilon\} & \text{if } a = 1 \\ \emptyset & \text{if } a = 0 \end{cases}$$

and extended homomorphically over all elements of the term algebra (called T_Σ). The definition of the residuals over the algebra of regular sets (see [7]) is given by

$$X/Y = \{z \in \Sigma^* \mid (\forall y \in Y) \, zy \in X\} \quad Y \backslash X = \{z \in \Sigma^* \mid (\forall y \in Y) \, yz \in X\}$$

where zy is the concatenation of two strings. Thus the laws (7)-(10) hold under the interpretation R_Σ are theorems in the algebra of regular sets.

In fact, AL_G^* is sound and complete for the algebra of regular sets under the standard interpretation [14]. This means that the results concerning the derivative of regular expressions [45] and those concerning the factors [5], in the context of regular language, hold in this algebra. That being said, Moor *et al.* [6] have shown how to determine if a regular expression is less than or equal to another using the factor matrix [15]. So we can use the procedure introduced by Moor to determine if an inequality holds. However, the complexity of this procedure [18] is exponential in the number of factors of an expression and not quite intuitive. For these reasons we present another technique for deciding inequalities in our algebra. At first sight, the complexity of our technique *seems* to be better than those of Moor *et al.* but we do not know for sure yet.

3.1 Comparison of Elements

The basic idea of our method is to reduce the comparison of any two elements to a comparison of an element to 1. Using (5), $x \leq y$ can be rewritten as $1 \leq y/x$ or $1 \leq x \backslash y$. Thus, the procedure is divided in two steps. First, we have to compute $x \backslash y$ and, second, we have to check if the result is greater than or equal to 1. Computing $x \backslash y$ can be done in a straightforward way by applying various laws and making use of the Theorem 1.

Theorem 1 (Finite division). *In any action lattice (with universe A) for which the laws (7) to (10) hold, for all $X, Y \in A$, there exists $j \in \mathbb{N}$ such that $X/Y^j = X/Y^{j+1}$.*

Since an action lattice for which the laws (7) to (10) hold is complete for the algebra of regular sets, Theorem 1 is a basic consequence of Theorem 5.2 in [4]. We can also prove Theorem 1 directly without using the fact that the considered algebra is complete (see [14] for details).

However, computing $x \backslash y$ in a straightforward way is difficult to implement and, in the end, it is rather equivalent to the technique develop by Moor *et al.* [6].

So instead of this, to compute $x \setminus y$, we will use the fact that any expression of the term algebra can be represented as an equational system. These equational systems can be seen as the regular grammar representing the expression of the term algebra. In [11] Kozen use this correspondence to prove that his axiomatization is complete for the algebra of regular. Here we will use a slightly different but equivalent representation of an expression as an equational system.

Let $LinEq$ be the set of all equational systems of the form given by (11) where the entries of the matrix A and B are elements of a fixed Kleene Algebra.

$$X = AX + B \quad (11)$$

The i^{th} row and j^{th} column of a matrix A are respectively $A_{i, \cdot}$ and $A_{\cdot, j}$ while its elements are designated by $a_{i,j}$. We define $\mathbf{0}$ as a matrix for which its entries are 0 and $\mathbf{1}$ as a matrix for which its entries are 1. Let $S \triangleq X = AX + B$ where X is a matrix of size $n \times 1$ and $S' \triangleq X' = A'X' + B'$ where X' is a matrix of size $m \times 1$ be two equational systems. Thus we define the sum, the product and the star of an equational system representing an expression as follows:

$$S +_E S' \triangleq Y = \left[\begin{array}{c|c|c} \mathbf{0} & A_{1,\cdot} & A'_{1,\cdot} \\ \hline \mathbf{0} & A & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & A' \end{array} \right] Y + \left[\begin{array}{c} B_{1,\cdot} + B'_{1,\cdot} \\ \hline B \\ \hline B' \end{array} \right] \quad (12)$$

$$S \cdot_E S' \triangleq Y = \left[\begin{array}{c|c} A & B | \mathbf{0} \\ \hline \mathbf{0} & A' \end{array} \right] Y + \left[\begin{array}{c} \mathbf{0} \\ \hline B' \end{array} \right] \quad (13)$$

$$S^{*E} \triangleq Y = \left[\begin{array}{c|c} \mathbf{0} & A_{1,\cdot} \\ \hline \mathbf{0} & [B \ B \ \dots \ B] \left[\begin{array}{c} A_{1,\cdot} \\ A_{1,\cdot} \\ \vdots \\ A_{1,\cdot} \end{array} \right] + A \end{array} \right] Y + \left[\begin{array}{c} \mathbf{1} \\ \hline B \end{array} \right] \quad (14)$$

where Y is a new matrix of variables of matching size. One might noticed that these operations are an algebraic equivalent of the proofs that the regular sets are closed under union, concatenation and transitive reflexive closure by construction of a finite state automaton.

The interpretation of an expression of the term algebra over an equational system, $S_\Sigma : T_\Sigma \mapsto LinEq$, is defined by:

$$S_\Sigma(a) \triangleq \begin{cases} X = \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix} X + \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } a \in \Sigma \\ X = [0]X + [0] & \text{if } a = 0 \\ X = [0]X + [1] & \text{if } a = 1 \end{cases}$$

and extended homomorphically over all elements. Using the correspondence between our representation and Kozen's representation of an expression [11, Lemma 15], it is easy to prove Corollary 2 [14].

Corollary 2. *Let e be an expression of the term algebra of a Kleene algebra then we have that*

$$e = [1 \ 0 \ \dots \ 0] A^* B$$

where A and B are the matrices obtained by computing $S_\Sigma(e)$.

Thus instead of reasoning in a Kleene algebra we can reason in an equational system representing the element of this algebra. The idea now is to define new operations over equational systems such that Corollary 2 can be extended to every expression of the term algebra of $AL_{\mathcal{G}}^*$.

The meet of equational systems is defined in the following way:

$$\begin{aligned} S \sqcap_E S' &\triangleq Y = \mathbf{A}Y + \mathbf{B} \\ \mathbf{A} \text{ is a matrix of size } nm \times nm \text{ and } \mathbf{a}_{i,j} &= a_{\lceil \frac{j}{m} \rceil, \lceil \frac{j}{m} \rceil} \sqcap a'_{i \bmod m, j \bmod m} \\ \mathbf{B} \text{ is a matrix of size } nm \times 1 \text{ and } \mathbf{b}_{i,1} &= b_{\lceil \frac{i}{m} \rceil, 1} \sqcap b'_{i \bmod m, 1} \end{aligned} \quad (15)$$

To compute the meet between entries of the matrices A and A' or B and B' we use the fact that, by construction, such entries are either 0, 1, $a \in \mathcal{G}$ or a sum of those and the following properties of $AL_{\mathcal{G}}^*$ (see [14] for the proofs) with universe A :

$$(x + y) \sqcap z = x \sqcap z + y \sqcap z \quad x, y, z \in A \quad (16)$$

$$(x \sqcap y) = 0 \quad x, y \in \{0, 1\} \cup \mathcal{G} \text{ and } x \neq y \quad (17)$$

For the residuals we only define the operation $\backslash_E$ on equational systems since $/_E$ can be defined from $\backslash_E$ in $AL_{\mathcal{G}}^*$ [14]. Hereafter, we define $\backslash_E$ by extending Corollary 2 (by structural induction) to all elements in $AL_{\mathcal{G}}^*$. So, we present the inductive case for the operator $\backslash_E$. Our induction hypotheses are that $S_{\Sigma}(e_1) \triangleq Y = AY + B$, where Y is a matrix of size $n \times 1$, and that $S_{\Sigma}(e_2) \triangleq X = CX + D$, where X is a matrix of size $m \times 1$, such that $e_1 = [1 \ 0 \ \dots \ 0]A^*B$ and $e_2 = [1 \ 0 \ \dots \ 0]C^*D$. We want to prove that $S_{\Sigma}(e_1 \backslash e_2) \triangleq Z = EZ + F$ such that $e_1 \backslash e_2 = [1 \ 0 \ \dots \ 0]E^*F$ and $S_{\Sigma}(e_1 \backslash e_2) = S_{\Sigma}(e_1) \backslash_E S_{\Sigma}(e_2)$.

From our induction hypotheses we know that $S_{\Sigma}(e_1)$ is a representation of e_1 as an equational system. More precisely, having that $e_1 = [1 \ 0 \ \dots \ 0]A^*B$ is equivalent to saying that e_1 is associated with the first row of Y , i.e. $Y_{1,.} = e_1$. The same is true for e_2 , i.e. $X_{1,.} = e_2$. Thus $e_1 \backslash e_2 = Y_{1,.} \backslash X_{1,.}$. So, instead of considering the explicit definition of e_1 and e_2 to compute $e_1 \backslash e_2$, we can consider the implicit definition of $Y_{1,.}$ and $X_{1,.}$ given by the equational systems $S_{\Sigma}(e_1)$ and $S_{\Sigma}(e_2)$. These implicit definitions of $Y_{1,.}$ and $X_{1,.}$ are respectively $Y_{1,.} = A_{1,.}Y + B_{1,.}$ and $X_{1,.} = C_{1,.}X + D_{1,.}$. Thus we have that:

$$Y_{1,.} \backslash X_{1,.} = (A_{1,.}Y + B_{1,.}) \backslash (C_{1,.}X + D_{1,.}) \quad (18)$$

Recalling that by construction, using S_{Σ} , the entries of the matrices A, B, C, D are either 0, 1, $a \in \mathcal{G}$ or a sum of those we can apply the laws of $AL_{\mathcal{G}}^*$ in order to simplify the right part of (18) so that we have:

$$Y_{1,.} \backslash X_{1,.} = \left(\bigcap_{1 \leq i \leq n} Y_{i,.} \backslash \left(\sum_{j' \in J_i} X_{j',.} \right) \right) \sqcap b_{1,1} \backslash X_{1,.}$$

where for each i $J_i \subseteq \{x : 1 \leq x \leq m\}$.

That being said, to compute $Y_{1..} \setminus X_{1..}$ we still have to compute

$$Y_{i..} \setminus \left(\sum_{j \in J_i} X_{j..} \right)$$

for each i . To do so we proceed in the same way as we did for $Y_{1..} \setminus X_{1..}$ by replacing the $Y_{i..}$ and $X_{j..}$ by their implicit definition and by simplifying the resulting terms. Once this is done we obtain an equational system defining $Y_{1..} \setminus X_{1..}$ where $Y_{i..} \setminus \left(\sum_{j \in J_i} X_{j..} \right)$ correspond to variables, say W_1 to W_k where W_1 is associated with $Y_{1..} \setminus X_{1..}$. To find the solution of this new equational system we apply the following algorithm:

1. Replace the definition of variables of the following form $W_i = W_i \sqcap V$ by $Z_i = V$ where V is a meet of variables.
2. In the definition of W_1 , select a variable. Replace this variable by its definition everywhere in the equational system. In other words, suppress all occurrences of this variable.
3. Repeat steps 1 and 2 until there is no more variables in the definition of W_1 .

The solution will then be of the form $Y_{1..} \setminus X_{1..} = \sum_{j \in J'} X_{j..}$. However, we know that the explicit definition of $X_{j..}$ is $U_j C^* D$, where U_i is a matrix of size $1 \times m$ where all its entries are equal to 0 except for $u_{1,i}$ which is equal to 1. Thus, $e_1 \setminus e_2 = Y_{1..} \setminus X_{1..} = \sum_{j \in J'} U_j C^* D$. Moreover, from $S_\Sigma(e_2)$ we can construct equational systems such that $U_j C^* D = [1 \ 0 \ \dots \ 0] C_j^* D_j$, say S_j for all $j \in J'$. This is done by swapping the first and the j^{th} rows of C and D and swapping the first and the j^{th} columns of C . It is now possible to define $\setminus_E$:

$$S \setminus_E S' \triangleq S_{j_1} +_E S_{j_2} +_E \dots +_E S_{j_l}$$

where j_i correspond to the i^{th} element of J' and $l = |J'|$.

4 An Application to Model Checking

One of our first goal to study the resolution of equations in Kleene algebras was to apply it to program analysis. As a step in that direction, we have been able to reduce the model checking of a restricted version of the linear μ -calculus over finite traces to a comparison of elements in a Kleene algebra. The choice of the linear μ -calculus is not arbitrary. It is based on the fact that each formula of the linear μ -calculus is equivalent to an ω -regular expression [3], and the fact that the algebra of ω -regular sets is a model of ω -algebra [2]. An ω -algebra is a Kleene algebra augmented with a unary operator ω . Intuitively, x^ω means that the action x is done infinitely often.

That being said, our model checking algorithm is equivalent to the one proposed by Moor *et al.* [6] in respect to programs and properties that we can verify. However, with our approach we are able to prove that the model checking of a

restricted version of the linear μ -calculus can be done in an algebraical way using Kleene algebras.

Model checking is done as follows. First, we translate the program and the formula into elements of the term algebra. Then we check if the translation of the program is less than or equal to the translation of the formula. This verification is done by using the method developed in the previous section. More precisely we are interested in verifying whether

$$\lceil P \rceil \leq \|\phi\|$$

is valid or not, where $\lceil P \rceil$ and $\|\phi\|$ are respectively the translation of the program P and the translation of the formula ϕ in the term algebra of action lattices for which the laws (7) to (10) hold. This inequality is further transformed into equational systems in respect of the method presented in Sect. 3.1.

Translation of a Formula. The logic considered is a restricted version of the linear μ -calculus that will be noted by L . The syntax is given by:

$$\phi ::= \text{tt} \mid \text{ff} \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid @a\phi \mid \text{eventually}(\phi)$$

where “ a ” is an action and $\text{eventually}(\phi)$ corresponds to $\mu Z. \phi \vee \odot Z$ in the linear μ -calculus, where $\odot$ means any action. In term of an action lattice, an action “ a ” is an element of $\mathcal{G}$ and $\odot$ corresponds to $\sum_{x \in \mathcal{G}} x$. For this syntax we have come up with a simple translation function:

$$\begin{aligned} \|\text{tt}\| &= \infty \\ \|\text{ff}\| &= 0 \\ \|\phi_1 \vee \phi_2\| &= \|\phi_1\| + \|\phi_2\| \\ \|\phi_1 \wedge \phi_2\| &= \|\phi_1\| \sqcap \|\phi_2\| \\ \|@a\phi\| &= a \cdot \|\phi\| \\ \|\text{eventually}(\phi)\| &= \infty \cdot \|\phi\| \end{aligned}$$

Given this translation function one may think that it is easy to extend it to the entire linear μ -calculus. However, the introduction of negation in the logic gives rise to many problems. We will come back on extending the translation function to the full linear μ -calculus in Sect. 5.

Then we translate the resulting expression in term of an equational system using the interpretation S_Σ .

Translation of a Program. Formulas of the linear μ -calculus are interpreted over infinite traces generated by a labelled Kripke structure. A labelled Kripke structure is a 6-tuple $(S, P, \text{AP}, \delta, \gamma, \text{Init})$ such that S is a finite set of states, P is a finite set of actions, AP is a finite set of atomic propositions, $\delta : S \times P \rightarrow 2^S$ is a transition function, $\gamma : S \rightarrow 2^{\text{AP}}$ is a labelling function and Init is the set of initial states. However, since we have restricted ourselves to finite traces in a logic without atomic propositions, labelled Kripke structures become a special case of non-deterministic finite state automata. Consequently, they can also be expressed by an equivalent deterministic finite state automaton. So we just have

to consider the deterministic finite state automaton represented as an equational system.

Theorem 2. *For any program P expressed as a finite state automaton and any formula ϕ of the logic L , we have:*

$$P \models \phi \leftrightarrow \lceil P \rceil \leq \|\phi\| .$$

The proof of this theorem is given in [14].

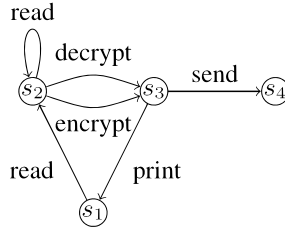
4.1 Example

Here we give a complete example presenting how model checking over the logic L can be done using AL_G^* . The property we want to verify is the following: “No information can be sent on the network after reading a file”. This property is expressed, in linear μ -calculus, by the formula

$$\neg \text{eventually}(\textcircled{r} \text{ eventually}(\textcircled{s} tt)) \quad (19)$$

where $r \triangleq \text{read}$ and $s \triangleq \text{send}$. However, instead of verifying directly this property we will rather consider its positive form. This positive form corresponds to the property that a *read* is followed by a *send*. So, any program that satisfies the positive form cannot satisfies (19) and *vice-versa*.

The program Pgm that will be considered is given by the following finite state automaton



which means that when a file is read successfully it is decrypted or encrypted before either printing it and start all over again or sending it on the network. Here the initial and final states of the program are respectively s_1 and s_4 .

Translation of the Program. Hereafter, we show how the program Pgm is translated to an equational system. We avoid many details of the computation and leave it to the reader (refer to [11]). The resulting equational system is:

$$Pgm \triangleq \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} 0 & r & 0 & 0 \\ 0 & r & d + e & 0 \\ p & 0 & 0 & s \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

where $r \triangleq \text{read}$, $p \triangleq \text{print}$, $d \triangleq \text{decrypt}$, $e \triangleq \text{encrypt}$ and $s \triangleq \text{send}$.

Translation of the Formula. The translation of the positive form of (19) is straightforward using the translation function:

$$\begin{aligned}
 \|\text{eventually}(\textcircled{r} \text{ eventually}(\textcircled{s} tt))\| &= \infty \cdot \|\textcircled{r} \text{ eventually}(\textcircled{s} tt)\| \\
 &= \infty \cdot r \cdot \|\text{eventually}(\textcircled{s} tt)\| \\
 &= \infty \cdot r \cdot \infty \cdot \|\textcircled{s} tt\| \\
 &= \infty \cdot r \cdot \infty \cdot s \cdot \|tt\| \\
 &= \infty \cdot r \cdot \infty \cdot s \cdot \infty \\
 &= \infty r \infty s \infty
 \end{aligned}$$

Applying S_{Σ} to $\infty r \infty s \infty$, and simplifying it, we obtain the following equational system:

$$Pr \triangleq \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} r + d + e + p + s & r & 0 \\ 0 & r + d + e + p + s & s \\ 0 & 0 & r + d + e + p + s \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Verification. The two equational systems representing the program and the formula are used to verify if the inequality

$$(rr^*(d+e)p)^*rr^*(d+e)s \leq \infty r \infty s \infty$$

is valid. Applying the method presented in Sect. 3.1 we first construct the new equational system which is equal to:

$$\begin{aligned}
 Y_1 \setminus X_1 &= Y_2 \setminus (X_1 + X_2) \\
 Y_2 \setminus (X_1 + X_2) &= Y_2 \setminus (X_1 + X_2) \sqcap Y_3 \setminus (X_1 + X_2) \\
 Y_3 \setminus (X_1 + X_2) &= Y_1 \setminus (X_1 + X_2) \sqcap Y_4 \setminus (X_1 + X_2 + X_3) \\
 Y_1 \setminus (X_1 + X_2) &= Y_2 \setminus (X_1 + X_2) \\
 Y_4 \setminus (X_1 + X_2 + X_3) &= X_1 + X_2 + X_3
 \end{aligned}$$

The solution of this system is:

$$X_1 + X_2 + X_3.$$

After simplification, the equational system corresponding to this solution is :

$$X = AX + B = [r + d + e + p + s]X + [1]$$

Since $b_{1,1} = 1$ we know that the inequality $1 \leq (rr^*(d+e)p)^*rr^*(d+e)s \setminus \infty r \infty s \infty$ is valid. Thus we proved that the program *Pgm* does not satisfy the property given in (19).

5 Future Work

In this paper, we presented a method for solving a particular class of linear equations in a subclass of *-continuous action lattices. Thus, we are far from being able to solve equations in any Kleene algebra. However, one should rather see this work as being a starting point for extending the resolution of equations

to larger subclasses of Kleene algebras. In that perspective, Sect. 5.1 surveys some of the possible extensions of this work to other classes of equations and classes of algebras.

Moreover, we showed how model checking can be done by algebraic handling based on the method developed for solving linear equations. However, the expressivity of the logic used is limited and this makes it almost impossible to apply it in real cases. In order to be able to do algebraic model checking with a more expressive logic we are working on the extension of the logic L to the entire linear μ -calculus. Section 5.2 presents the stages to be crossed to achieve this goal.

5.1 Equations in Kleene Algebra

First of all, in order to solve any linear equations we have to be able to solve equations in which the variable appears on both sides of the equality sign. Other interesting classes of equations are the non-linear ones. For example, such non-linear equations are

$$\infty X^2 \infty = acb + abac \qquad XaXbX = cacbc \qquad aX^2 + bX + c = d$$

where $a, b, c, d \in A$, the universe of a Kleene algebra. However, while surveying these equations we soon discovered that solving them is far from obvious. That been said, with non-linear equations we can specify properties which *seem* to be unexpressed by any other means that the authors are acknowledge of. For example, let say that P is an expression of a Kleene algebra representing a program. An interesting property to verify is if it begins and ends with the “same block of actions” where this block of actions is not defined by the user at all. To verify such a property we would have to determine whether the inequation

$$P \leq X \infty X$$

has a solution or not, where X is a variable. Being able to verify this kind of property would be an important step forward in model checking.

In the literature related to Kleene algebra there is a lot of work done on Kleene algebra with tests. It has been used in number of applications to reason on computer programs. Thus it is an algebra where it would be interesting to solve equations.

Lastly, it would be interesting to develop a method of resolution on algebras with laws not as strong as laws (7) to (10). This becomes particularly useful when we reason on the equivalence of programs calling upon a first-order logic.

5.2 Toward the Linear μ -Calculus

In Sect. 4, we have presented how model checking of the logic L over finite traces could be done by algebraic handling. Since the expressivity of this logic

is rather limited we wish to extend the algebraic model checking to the entire linear μ -calculus. The syntax of this logic is given by:

$$\phi := p \mid Z \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid @\phi \mid \mu Z.\phi$$

where “ p ” is an atomic proposition, “ a ” is an atomic action and Z is a variable. Moreover, the semantics of the linear μ -calculus is defined on infinite traces.

First of all, since there is atomic propositions in the linear μ -calculus we will have to translate a formula in Kleene algebra with tests. In order to be able to reason on this class of algebras, we will proceed in a similar way as we did for Kleene algebra without tests. We will restrict ourselves to algebras in which there is laws allowing us to compute the division of two elements. These laws will be based on theorems in the algebra of regular sets of guarded strings.

Then we will have to translate the negation. In a Kleene algebra there is no such thing as the negation. This means that we will have to find an equivalent positive form of a negated expression. However, this is not the only problem related to the introduction of the negation in a logic. For instance, many problems arise from the dual form of formulas.

Moreover, to consider formulas such as $\mu Z.\phi$ we have to solve linear equations in which the variable appear on both sides of the equality sign. The intuition behind it is that $\mu Z.\phi$ is the least value such that $Z = \phi(Z)$. Thus by definition of the translation function, $\|Z\| = \|\phi(Z)\|$ is a Kleene algebra equation equivalent to $Z = \phi(Z)$. So in order to find the least value of $\|Z\| = \|\phi(Z)\|$ we need to solve equations in which the variable appears on both sides of the equality sign.

Finally, by considering ω -algebra, we will no longer be limited to finite traces. This means that, to be able to consider the entire linear μ -calculus, we have to be able to solve linear equations in ω -algebra with tests.

6 Conclusion

In this paper, we developed a method for solving linear equations in which the variable appears on one side of the equality sign over a $*$ -continuous action lattice. The choice of this kind of equations and algebraic structures was a consequence of various constraints we observed. We are now looking to extend this work to be able to solve more equations and to find other applications of it. As a step in that direction, we start working on solving linear equations in an ω -algebra with tests. Model checking is our first motivation behind this work since we want to find what is missing in a program to satisfy a particular property. However, other possible applications of the resolution of equations might be found in program equivalence as in the synthesis of controllers.

Acknowledgements

We are grateful to Jules Desharnais for his comments and suggestions. Also we are thankful to the anonymous referees which reviewed this article helping make it better.

References

1. Backhouse, R.C.: Regular algebra applied to language problems. *Journal of Logic and Algebraic Programming* 66(2), 71–111 (2006)
2. Bolduc, C.: Oméga Algèbre: Théorie et application en vérification de programmes. Master's thesis, Université Laval (2006)
3. Bradfield, J., Stirling, C.: Modal logics and mu-calculi: an introduction (2001)
4. Brzozowski, J.A.: Derivatives of regular expressions. *J. ACM* 11(4), 481–494 (1964)
5. Conway, J.H.: *Regular Algebra and Finite Machines*. Chapman and Hall, Boca Raton (1971)
6. de Moor, O., Drape, S., Lacey, D., Sittampalam, G.: Incremental program analysis via language factors (submitted for publication, 2002)
7. Höfner, P.: From Sequential Algebra to Kleene Algebra: Interval Modalities and Duration Calculus. Technical report, University of Augsburg (2005)
8. Jipsen, P.: From semirings to residuated kleene lattices. *Studia Logica* 76(2), 291–303 (2004)
9. Jipsen, P., Tsinakis, C.: A Survey of Residuated Lattices. In: Martinez, J. (ed.) *Ordered Algebraic Structures*, pp. 19–56. Kluwer Academic Publishers, Dordrecht (2002)
10. Kozen, D.: On Kleene algebras and closed semirings. In: Rovan, B. (ed.) *MFCS 1990. LNCS, vol. 452*, pp. 26–47. Springer, Heidelberg (1990)
11. Kozen, D.: A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Information and Computation* 110, 366–390 (1994)
12. Kozen, D.: On action algebras. In: van Eijck, J., Visser, A. (eds.) *Logic and Information Flow*, pp. 78–88. MIT Press, Cambridge (1994)
13. Kozen, D.: Kleene Algebra with Tests. *ACM Transactions on Programming Languages and Systems* 19(3), 427–443 (1997)
14. Ktari, B., Lajeunesse-Robert, F., Bolduc, C.: Solving Linear Equations in *-continuous Action Lattices (Extended Version). Technical Report DIUL-RR-0801, Département d'informatique et de génie logiciel, Université Laval, p. 30 (2008)
15. Lajeunesse-Robert, F., Ktari, B.: Toward Solving Equations in Kleene Algebra. In: *Proceedings of the 6th international Conference on Software Methodologies, Tools and Techniques (SoMeT 2007)*, Roma, Italy, p. 20. IOS Press, Amsterdam (2007)
16. Möller, B.: Residuals and Detachment. Technical report, University of Augsburg (2005)
17. Nutt, W.: Unification in Monoidal Theories is Solving Linear Equations over Semirings. Technical Report RR-92-01, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Erwin-Schrödinger Strasse, Postfach 2080, 67608 Kaiserslautern, Germany (1992)
18. Sittampalam, G., de Moor, O., Larsen, K.F.: Incremental execution of transformation specifications (2004)
19. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time(Preliminary Report). In: *STOC 1973: Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 1–9. ACM Press, New York (1973)
20. Suikang, D.: Proseminar Kleene Algebra und Regular Expressions (May 2004)

Reactive Probabilistic Programs and Refinement Algebra

L.A. Meinicke and K. Solin

Åbo Akademi, Finland

larissa.meinicke@abo.fi, kim.solin@abo.fi

Abstract. A trace semantics is given for a probabilistic reactive language which is capable of modelling probabilistic action systems. It is shown that reactive probabilistic programs with the trace semantics form a general refinement algebra. The abstract-algebraic characterisation means that the proofs of earlier-established transformation rules can be reused for probabilistic action systems with trace semantics.

1 Introduction

Refinement-algebraic reasoning has been used to verify transformation rules for probabilistic programs with an input-output (or sequential) semantics [7,6]. It has not, however, been applied to reactive probabilistic programs: programs in which behaviour over time, as well as input-output behaviour is visible. In this paper we define a reactive probabilistic program language with trace semantics, and we show that these programs form a general refinement algebra.

Our reactive probabilistic programs consist of atomic actions, which are expressed using probabilistic guarded commands, and guards and assertions, which may be composed using sequential composition, discrete probabilistic choice, demonic nondeterministic choice, and weak and strong iteration operators. The language is capable of modelling probabilistic action systems [12,16], a construct which may be used to express concurrent probabilistic systems. The trace semantics we present for the language may be seen to generalise the non-probabilistic action system trace semantics of Back and von Wright [1]. It also bears similarities to relational (non-reactive) probabilistic program semantics such as that in [10], which we use to describe the input-output behaviour of atomic actions.

The set of reactive programs over a fixed state space interpreted according to our trace semantics forms a general refinement algebra with enabledness and termination. This provides a simple but important link between non-reactive and reactive probabilistic program models, and means that the algebraic theory developed for general refinement algebra can immediately be applied to programs with our trace semantics, yielding for example two transformation rules for reactive probabilistic action systems. There has been other recent interest in applying abstract-algebraic methods to (non-probabilistic) programs with trace semantics. For example, Möller has shown how a simple trace model forms a lazy Kleene algebra [9], and this has been taken further in work by Desharnais,

Möller and Struth [2], and in work by Höfner and Struth [5]. Our work can be seen as a contribution to this line of work.

The paper proceeds according to the following outline. The probabilistic reactive language is introduced in Sect. 2, and its trace model is defined in Sect. 3. In Sect. 4 we define the trace semantics of the probabilistic reactive language primitives and operators in the reactive probabilistic model. We show how to interpret the probabilistic reactive programs as a general refinement algebra with enabledness and termination operators in Sect. 5, and we identify some extra algebraic rules that apply to the reactive programs. Applications of the algebra are also discussed in this last section.

2 The Probabilistic Reactive Language

Our probabilistic reactive language may be used to express reactive probabilistic programs at the level of code. It contains distinguished programs **abort**, **magic** and **skip**, in addition to atomic actions, $\langle A \rangle$, guards, $[g]$, and assertions, $\{g\}$, sequential composition, $;$, discrete probabilistic choice, $_p\oplus$, demonic nondeterministic choice, $\sqcap$, weak, $*$, and strong, $^\omega$, iteration operators. The body A of each atomic action $\langle A \rangle$, is described by a probabilistic guarded command,

$$A \triangleq \text{abort} \mid \text{magic} \mid \text{skip} \mid \{g\} \mid [g] \mid x := E \mid A_p \oplus A \mid A \sqcap A \mid A; A \mid A^* \mid A^\omega,$$

where x is a variable name, g is a predicate on the state space, E is an expression on the state space, and p is a probability. The language operators other than the atomic action language construct are overloaded since they are also defined for the bodies of atomic actions. The unary operators $^\omega$ and $*$ have the highest precedence, followed by the binary operators $;$, $\sqcap$, and then $\oplus$.

Program abortion is used in our reactive model to represent catastrophic failure, that cannot be averted by subsequent commands, and that cannot change the past. So-called miraculous behaviour is unimplementable, but is required for modelling guards. Unlike program abortion, the execution of **magic** may actually constrain the past, turning (non-aborting and non-infinite) behaviour which has already occurred to magic, effectively preventing a reactive program from taking a path which cannot be executed.¹ Guards are an important primitive construct. They may, for instance, be used to model more complex statements such as conditionals (e.g., if g then S else $T \triangleq [g]; S \sqcap [\neg g]; T$). The ability to express conditions like that in the if-statement *as programs themselves* simplifies the algebraic treatment of reactive probabilistic programs.

3 Semantics

First we describe the sequential program semantics that we use to describe the action bodies, and then we define the trace semantics of the reactive probabilistic programs.

¹ We treat both aborting and miraculous program behaviour in much the same way they are treated in the real-time refinement calculus [3].

Let d_1 and d_2 be distributions of type $\overline{\Sigma_\top}$; p be a constant in $[0..1]$; Γ be a subset of $\Sigma_\top$; and σ be an element of $\Sigma_\top$.

$\overline{\Sigma_\top}$	$\triangleq$	$\{d : \Sigma_\top \rightarrow [0..1] \mid (\sum_{\sigma \in \Sigma_\top} d.\sigma) \leq 1\}$
$d_1.\Gamma$	$\triangleq$	$(\sum_{\gamma \in \Gamma} d_1.\gamma)$
$d_1 \leq d_2$	$\triangleq$	$(\forall \Gamma \subseteq \Sigma \bullet d_1.\top + d_1.\Gamma \leq d_2.\top + d_2.\Gamma)$
$d_{1p} \oplus d_2$	$\triangleq$	$(\lambda \sigma \in \Sigma_\top \bullet p \times d_1.\sigma + (1-p) \times d_2.\sigma)$
$\overline{\sigma}$	$\triangleq$	$(\lambda \sigma' \in \Sigma_\top \bullet \sigma' = \sigma)$

Fig. 1. Distribution notation

3.1 Semantics of Actions

Several similar relational models for sequential probabilistic programs have been proposed. We use the model defined by McIver and Morgan [10] which facilitates the expression of miraculous program behaviour.

Let $\Sigma_\top$ be a state space Σ extended with a special state $\top$, which is used for representing miraculous program behaviour. The set of *discrete sub-probability distributions* over $\Sigma_\top$, $\overline{\Sigma_\top}$, is defined in Fig. 1, along with the ordering on distributions, $\leq$, and a number of operators. For any distribution, $d \in \overline{\Sigma_\top}$, the unallocated probability $1 - d.\Sigma_\top$ will be used to represent the probability of *aborting*, while $d.\top$ will represent the probability of behaving *miraculously*. Informally, we have that $d_1 \leq d_2$ if d_1 can be transformed into d_2 by possibly (a) reducing the probability of abortion by increasing the probability of reaching any state, and/or by (b) replacing the probability of reaching real-states with miraculous behaviour. We refer to the greatest distribution, $\overline{\top}$, as $d_\top$.

A set of distributions $D \subseteq \overline{\Sigma_\top}$ is *up-closed* if $(\forall d_1 \in D \bullet (\forall d_2 \in \overline{\Sigma_\top} \bullet d_1 \leq d_2 \Rightarrow d_2 \in D))$ and *convex-closed* if $(\forall p \in [0..1], d_1, d_2 \in D \bullet (\exists d_3 \in D \bullet d_3 = d_{1p} \oplus d_2))$. A set of distributions is *healthy* if it is non-empty, up- and convex-closed, and *Cauchy-closed*, meaning that it is closed in the usual Euclidean sense [10]. The up-closure of a set D of distributions is written $\lceil D \rceil$.

Programs on a fixed state space Σ , which is assumed to be a mapping from variable names to values, are modelled by *top preserving* functions from $\Sigma_\top$ to healthy sets of discrete probability distributions over $\Sigma_\top$. A function is *top preserving* if it maps state $\top$ to the set which contains only the top distribution, $d_\top$. This constrains programs so that they cannot leave the miraculous state $\top$. As the refinement relation between probabilistic programs is defined by subset inclusion, $A \sqsubseteq B \triangleq (\forall \sigma \in \Sigma \bullet B.\sigma \subseteq A.\sigma)$, the healthiness properties up- and convex-closure ensure that aborting behaviour may be refined in any way, miraculous behaviour may replace non-miraculous behaviour, and that demonic choices may be refined by probabilistic choices.

The relational semantics of the probabilistic guarded commands is listed in Fig. 2. The weak, $*$, and strong, $^\omega$, iteration operators are defined using greatest,

² This semantics is a slight adjustment of the semantics which appeared in [6]: it has been adjusted so that aborting behaviour is expressible.

Let Σ be a state space; g a predicate on Σ ; p a probability in $[0..1]$; x a variable in the domain of Σ ; E a function from the states in Σ to possible values of variable x in Σ ; A and B be sequential probabilistic programs on state space Σ ; and F a function of type $\Sigma_{\top} \rightarrow \overline{\Sigma}_{\top}$. For each guarded command A we define $A.\top = \{d_{\top}\}$, and for all other $\sigma_0 \in \Sigma$ we define

$\text{abort}.\sigma_0$	$\triangleq$	$\overline{\Sigma}_{\top}$
$\text{magic}.\sigma_0$	$\triangleq$	$\{d_{\top}\}$
$\text{skip}.\sigma_0$	$\triangleq$	$[\{\overline{\sigma_0}\}]$
$\{g\}.\sigma_0$	$\triangleq$	if $g.\sigma_0$ then $\text{skip}.\sigma_0$ else $\text{abort}.\sigma_0$
$[g].\sigma_0$	$\triangleq$	if $\underline{g}.\sigma_0$ then $\text{skip}.\sigma_0$ else $\text{magic}.\sigma_0$
$(x := E).\sigma_0$	$\triangleq$	$[\{\sigma_0[x \setminus E.\sigma_0]\}]$
$(A_p \oplus B).\sigma_0$	$\triangleq$	$\{d \in \overline{\Sigma}_{\top} \mid (\exists d_1 \in A.\sigma_0, d_2 \in B.\sigma_0 \bullet d = d_{1p} \oplus d_2)\}$
$(A \sqcap B).\sigma_0$	$\triangleq$	$\bigcup \{p \in [0..1] \bullet (A_p \oplus B).\sigma_0\}$
$(A; B).\sigma_0$	$\triangleq$	$\left\{ d \in \overline{\Sigma}_{\top} \mid (\exists d_1 \in A.\sigma_0, F : \Sigma_{\top} \rightarrow \overline{\Sigma}_{\top} \bullet \right.$ $\left. (\forall \sigma \in \Sigma_{\top} \bullet F.\sigma \in B.\sigma) \wedge F^{\dagger}.d_1 \leq d) \right\}$
$F^{\dagger}$	$\triangleq$	$(\lambda d \in \overline{\Sigma}_{\top} \bullet (\lambda \sigma \in \Sigma_{\top} \bullet \sum_{\gamma \in \Sigma_{\top}} F.\gamma.\sigma \times d.\gamma))$
A^*	$\triangleq$	$(\nu X \bullet A; X \sqcap \text{skip})$
A^{ω}	$\triangleq$	$(\mu X \bullet A; X \sqcap \text{skip})$

Fig. 2. Relational semantics for probabilistic guarded command language

ν , and least, μ , fixpoints over the set of programs with refinement ordering $\sqsubseteq$. These are well defined since this set of programs form a complete partial order with respect to $\sqsubseteq$.

3.2 Trace Semantics of Reactive Programs

The trace semantics of a reactive probabilistic program captures the possible behaviours of the program over time, where a behaviour over time is described by the states that are produced after the execution of each atomic action. Each possible behaviour over time, which we refer to as a *behaviour tree*, is branching, and not linear, since it describes a possible “probabilistic execution” of the program. To be more precise, given any finite sequence of states that might be produced, a behaviour tree describes a distribution of next states that may be reached from that point in one possible probabilistic execution.³ In an “actual” execution of a probabilistic reactive program, the probabilistic choices may be resolved according to the distributions used to describe them in a behaviour tree: this results in the production of a *behaviour*, which is a linear sequence of states.

We first express the notion of behaviours and behaviour trees formally, and then use these to define the semantics of the reactive probabilistic programs.

³ It is important that the distribution of next-states is dependent on the path taken to reach that point, and not just the last state in that path. If the distribution of next-states were dependent on the previous state alone, then the program would have to resolve all demonic nondeterministic choices from the same state in the same way.

Behaviours and Behaviour Trees. The set of *behaviours* on state space Σ is defined as

$$\text{beh}.\Sigma \triangleq (\text{seq}.\Sigma \cap \{\langle \rangle, \langle \neg \rangle\}) \cup (\text{iseq}.\Sigma) \cup \{\langle \top \rangle\},$$

which represents the set of all finite and infinite sequences of states from Σ , where the finite sequences may be terminated by the special termination state $\neg$, and $\langle \top \rangle$ is a magical behaviour. A behaviour b (which is not the magical behaviour $\langle \top \rangle$) is defined to be *terminating* if it is finite and its last state is $\neg$; *aborting* if it is finite and its last element is not $\neg$; and *nonterminating* if it is neither terminating nor aborting⁴

Given a state space Σ , the set of possible *behaviour trees* on Σ may be modeled as functions from finite sequences of states from Σ , denoted $\text{seq}.\Sigma$, to discrete distributions over the state space $\Sigma_{\top, \neg}$, which is a state space $\Sigma_{\top}$ extended with the special state $\neg$ representing termination. In symbols,

$$\text{behTree}.\Sigma \triangleq \{t : \text{seq}.\Sigma \rightarrow \overline{\Sigma_{\top, \neg}} \mid (\forall s \in \text{seq}.\Sigma \bullet \text{size}.s \geq 1 \Rightarrow t.s.\top = 0)\}.$$

Given a behaviour tree $t \in \text{behTree}.\Sigma$, and a finite sequence $s \in \text{seq}.\Sigma$, $t.s$ denotes the next state distribution of t after producing the sequence of states s . The probability of aborting after producing output sequence s is $1 - (t.s).\Sigma_{\neg}$. The probability of terminating after producing s is $t.s.\neg$. The probability that t will perform magic is $t.\langle \rangle.\top$: t may either reach the magical state immediately, after producing $\langle \rangle$, or not at all.

A part of a tree is said to be unreachable if it can only be reached by a tree edge with probability 0. The reachable part of a behaviour tree (the part we are interested in) may be succinctly described by its probability of producing any finite prefix of a behaviour⁵. For a behaviour tree t , and a finite behaviour prefix s , $\text{pExpt}.t.s$ denotes the probability that the deterministic behaviour tree t will produce behaviour prefix s . The value of $\text{pExpt}.t.s$ may be calculated simply by multiplying together the probabilities along the branches of t that are taken to produce s . Formally, for a behaviour tree $t \in \text{behTree}.\Sigma$, a finite behaviour prefix $s \in \text{seq}.\Sigma$, and a state $\sigma \in \Sigma_{\top, \neg}$, the function $\text{pExpt} : \text{behTree}.\Sigma \rightarrow (\text{seq}.\Sigma \cap \{\langle \rangle, \langle \neg \rangle, \langle \top \rangle\}) \rightarrow \mathbb{R}$ is defined by

$$\text{pExpt}.t.\langle \rangle \triangleq 1, \quad \text{pExpt}.t.(s \frown \langle \sigma \rangle) \triangleq \text{pExpt}.t.s \times t.s.\sigma.$$

⁴ The special state $\neg$ serves a similar role to the special *ok* variable in UTP [4]: it is used to distinguish terminated tree branches from aborted ones. Infinite tree branches are inherently *ok*. As we will see, finite branches which are not *ok* are not modified by sequential composition.

⁵ The reachable part of a behaviour tree, which is described by its probability of producing any finite behaviour prefix, can be used to construct a probability measure over (both finite *and* infinite) behaviours. This approach is taken, for instance, in the work of Segala [11]. We retain a simple tree-based definition of probabilistic behaviours, in preference to a measure theoretic definition, since our behaviour tree partial order would remain unchanged, and the extra theory is not required for this paper. It is sufficient to observe that our definition of a behaviour tree implicitly describes infinite, as well as finite, behaviours.

On sets $Q \subseteq (seq.\Sigma \cap \{\langle \rangle, \langle \dashv \rangle, \langle \top \rangle\})$, we overload this notation, writing $p\text{Expt}.t.Q$ to mean $\sum_{s \in Q} p\text{Expt}.t.s$.

The ordering between two behaviour trees t_1 and t_2 is then defined analogously to the ordering on distributions by $t_1 \leq t_2 \triangleq$

$$(\forall Q \subseteq \Sigma \cdot \text{indep}.Q \Rightarrow p\text{Expt}.t_1.\langle \top \rangle + p\text{Expt}.t_1.Q \leq p\text{Expt}.t_2.\langle \top \rangle + p\text{Expt}.t_2.Q),$$

where $\text{indep}.Q \triangleq (\forall s_1, s_2 \in Q \cdot s_1 \text{ not a prefix of } s_2 \text{ and } s_2 \text{ not a prefix of } s_1)$.

This partial ordering on behaviour trees only makes reference to the reachable parts of the trees. It states that one tree t_1 may be transformed into a greater tree by either extending its aborting branches or by removing branches, or parts of branches, and replacing them by miraculous behaviour. We refer to the greatest tree with respect to the refinement ordering as $t_\top$, where we have that $t_\top.\langle \rangle.\top = 1$, and the tree that terminates right away as $t_\dashv$, for which $t_\dashv.\langle \rangle.\dashv = 1$.

Given two behaviour trees t_1 and t_2 and a probability p in $[0, 1]$ we define the probabilistic combination of both trees, $t_{1p} \oplus t_2$, by the unique tree t_3 such that

$$(\forall s \in seq.\Sigma \cap \{\langle \rangle, \langle \dashv \rangle, \langle \top \rangle\} \cdot p\text{Expt}.t_3.s = p \times p\text{Expt}.t_1.s + (1 - p) \times p\text{Expt}.t_2.s).$$

Program Interpretation and Trace Refinement. We want to express the meaning of a reactive program S on a state space Σ as a mapping from states to sets of healthy behaviour trees. The healthiness conditions we require are that the set of behaviour trees be non-empty, up- and convex-closed (with respect to the ordering on behaviour trees). These conditions can be defined analogously to the conditions in Sect. 3.1. The up-closure of a set of trees D is written $\lceil D \rceil$.

Let $H \subseteq \text{behTree}.\Sigma$ be the set of all healthy behaviour trees. The set of *reactive programs* on state space Σ , React_Σ , is then the set of functions of type $\Sigma \rightarrow H$. From a given initial state σ , $S \in \text{React}_\Sigma$ may nondeterministically choose to behave according to any of one the healthy behaviour trees in $S.\sigma$. The possible behaviour trees produced by a reactive program are dependent on the initial state only, since the behaviour of atomic actions is only dependent on the initial state, and not on the history of execution. We define *refinement* between reactive programs S and T via subset inclusion of behaviour tree sets: $S \sqsubseteq T \triangleq (\forall \sigma \in \Sigma \cdot T.\sigma \subseteq S.\sigma)$. With respect to this refinement ordering, React_Σ forms a complete partial order.

Example 1. Fig. 3 shows a reactive program S , and one possible behaviour tree t_S from $S.\sigma_0$, for some σ_0 . In the graphical representation of the behaviour tree each node either represents a state, which is denoted by the value of variable x , the termination symbol $\dashv$, or the magical state $\top$. For each node, the weighted edges leaving the node define the probability of reaching different next states, given that the states along the path from the root to the node have been chosen. We can see that the terminating behaviour $\langle 1, \dashv \rangle$ is produced with probability $\frac{1}{8}$ if the reactive program chooses to behave like t_S .

$$S \triangleq (\langle x := 1 \rangle; (\text{skip}_{\frac{1}{4}} \oplus \langle x := 2 \rangle; \text{abort}))_{\frac{1}{2}} \oplus \text{skip}$$

$$t_S \triangleq$$

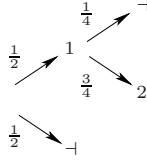


Fig. 3. Probabilistic reactive program S , and a behaviour tree t_S that may be produced by S from any initial state

4 Trace Semantics for the Reactive Language

Now that the trace model that we use to specify reactive probabilistic programs has been defined, we present the trace semantics of the reactive program primitives and operators, and we show how they may be used to specify probabilistic action systems.

The Primitive Commands. The semantics of the language primitives are defined in Fig. 4. Program **abort** is the least reactive program, which maps each initial state to the set of all possible behaviour trees, and **magic** is the top reactive program, which maps each state to the set containing the top behaviour tree, $t_{\top}$. Reactive program **skip** terminates right away, effectively performing “no action.” As we will see, it is the unit of sequential composition. As for sequential probabilistic programs, a guard $[g]$ acts like **skip** from states in which g holds, and **magic** otherwise. Likewise, an assertion $\{g\}$ skips from states in which predicate g holds, and behaves like **abort** from other states. Given an action A , and an initial state σ_0 , $\langle A \rangle.\sigma_0$ defines the set of behaviour trees that are constructed by atomically executing action A from σ_0 and then terminating. It is instructive to note the difference between the atomic action $\langle \text{skip} \rangle$ and the reactive program **skip**: the first performs a visible action, whereas the second does not. All of these primitive commands satisfy the healthiness conditions stated in Sect. 3.2; they produce sets of distributions which are non-empty, up- and convex-closed.

The Composition Operators. The composition operators are defined in Fig. 5; they are akin to their non-reactive counterparts from Sect. 3.1. The probabilistic and nondeterministic choice operators behave as expected.

The sequential composition operator is more complex. Given two reactive programs S and T , and an initial state σ_0 , each tree in $(S; T).\sigma_0$ is produced by taking tree t_1 from $S.\sigma_0$ and extending each of its terminating branches $s \hat{\ } \langle \cdot \rangle$ with a tree from $T.(last.(\langle \sigma_0 \rangle \hat{\ } s))$. Given a tree t_1 and a function $F : seq.\Sigma \rightarrow behTree.\Sigma$ which describes how each terminating branch of t_1 should be extended, function *extendTree* defines the tree that is produced by extending t_1 according to F .

Let $\sigma_0 \in \Sigma$; g be a predicate on Σ ; and A a sequential probabilistic program.

$\text{abort}.\sigma_0$	$\triangleq$	$\text{behTree}.\Sigma$
$\text{magic}.\sigma_0$	$\triangleq$	$\{t_\top\}$
$\text{skip}.\sigma_0$	$\triangleq$	$[\{t_\perp\}]$
$[g].\sigma_0$	$\triangleq$	if $\sigma_0 \in g$ then $\text{skip}.\sigma_0$ else $\text{magic}.\sigma_0$
$\{g\}.\sigma_0$	$\triangleq$	if $\sigma_0 \in g$ then $\text{skip}.\sigma_0$ else $\text{abort}.\sigma_0$
$\langle\!\langle A \rangle\!\rangle.\sigma_0$	$\triangleq$	$\left\{ \begin{array}{l} t \in \text{behTree}.\Sigma \mid (\exists t' \in \text{behTree}.\Sigma \bullet \\ (\exists d \in A.\sigma_0 \bullet (\forall \sigma \in \Sigma_\top \bullet t'.\langle\sigma\rangle.\sigma = d.\sigma)) \bullet \\ \wedge (\forall \sigma \in \Sigma \bullet t'.\langle\sigma\rangle.\perp = 1) \wedge t' \leq t) \end{array} \right\}$

Fig. 4. Reactive program primitives

Let $\sigma_0 \in \Sigma$; and S, T be probabilistic reactive programs; and $p \in [0..1]$.

$(S_p \oplus T).\sigma_0$	$\triangleq$	$\{t \in \text{behTree}.\Sigma \mid (\exists t_1 \in S.\sigma_0, t_2 \in T.\sigma_0 \bullet t = t_{1p} \oplus t_2)\}$
$(S \sqcap T).\sigma_0$	$\triangleq$	$\bigcup_{p \in [0,1]} (S_p \oplus T).\sigma_0$
$(S; T).\sigma_0$	$\triangleq$	$\{t \in \text{behTree}.\Sigma \mid (\exists t_1 \in S.\sigma_0, F : \text{seq}.\Sigma \rightarrow \text{behTree}.\Sigma \bullet$ $(\forall s \in \text{seq}.\Sigma \bullet F.s \in T.(last.(\langle\sigma_0\rangle \frown s)))$ $\wedge \text{extendTree}.(t_1, F) \leq t)\}$
T^ω	$\triangleq$	$(\mu X \bullet T; X \sqcap \text{skip})$
T^*	$\triangleq$	$(\nu X \bullet T; X \sqcap \text{skip})$

Fig. 5. Reactive program composition operators

We modularise function *extendTree* as follows: first we define a *concatTree* function that applied to a tree takes us outside the set of well-defined behaviour trees and then we define a *trim* function that appropriately trims the tree back into a behaviour tree. The *concatTree* function will be used to simply concatenate the tree. This function may produce a tree which has branches of length greater than one that end in the magical state $\top$. Such a tree is referred to as a relaxed tree, $\text{relaxbehTree}.\Sigma \triangleq \text{seq}.\Sigma \rightarrow \overline{\Sigma}_{\perp, \top}$. The role of the second function, *trim*, will be to prune the miraculous branches from the tree, collecting their probabilities together at the root.

Function $\text{concatTree} : (\text{behTree}.\Sigma \times \text{seq}.\Sigma \rightarrow \text{behTree}.\Sigma) \rightarrow \text{relaxbehTree}.\Sigma$ is defined via pExpt as follows: $\text{concatTree}.(t, F)$ is the unique relaxed behaviour tree such that

1. for any $s \in \text{seq}.\Sigma \cap \{\langle\cdot\rangle, \langle\top\rangle\}$

$$\begin{aligned} \text{pExpt}.(concatTree.(t, F)).s &= \text{pExpt}.t.s + \\ &(\sum_{\{s', s'' \in \text{seq}.\Sigma \mid s = s' \frown s'' \wedge \text{size}.s'' > 0\}} \text{pExpt}.t.(s' \frown \langle\cdot\rangle) \times \text{pExpt}.(F.s').s''), \end{aligned}$$

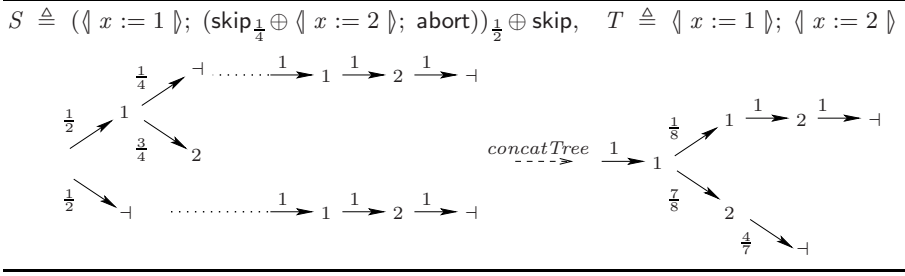


Fig. 6. Reactive programs S and T and the construction of one possible behaviour tree from $(S; T).\sigma_0$

2. and for any $s \in \text{seq}.\Sigma$ we define

$$\begin{aligned} & \text{pExpt.}(\text{concatTree.}(t, F)).(s \frown \langle - \rangle) = \\ & \left(\sum_{\{s', s'' \in \text{seq}.\Sigma \mid s = s' \frown s''\}} \text{pExpt.}t.(s' \frown \langle - \rangle) \times \text{pExpt.}(F.s').(s'' \frown \langle - \rangle) \right). \end{aligned}$$

We cannot include $\text{pExpt.}t.(s \frown \langle - \rangle)$ in this last sum since the branch $s \frown \langle - \rangle$ from t may be extended by the sequential composition so that it no longer terminates after producing the behaviour prefix s .

Given a tree $t_{\text{tmp}} \in \text{relaxbehTree}.\Sigma$ we again define *trim* via $\text{pExpt. trim.}(t_{\text{tmp}})$ is the unique behaviour tree t such that

1. For $s \in \text{seq}.\Sigma - \{\langle \rangle\}$,

$$\text{pExpt.}t.s = \text{pExpt.}t_{\text{tmp}}.s - \sum_{s' \in \{s' \in \text{seq}.\Sigma \mid s \text{ prefix of } s'\}} \text{pExpt.}t_{\text{tmp}}.s' \frown \langle \top \rangle,$$

2. for $s \in \text{seq}.\Sigma$, $\text{pExpt.}t.s \frown \langle - \rangle = \text{pExpt.}t_{\text{tmp}}.s \frown \langle - \rangle$, and

3. $\text{pExpt.}t.\langle \top \rangle = \sum_{s \in \text{seq}.\Sigma} \text{pExpt.}t_{\text{tmp}}.s \frown \langle \top \rangle$.

Such a tree always exists and only one such tree exists. The *extendTree* function can now be defined by $\text{extendTree.}(t, F) \triangleq \text{trim.}(\text{concatTree.}(t, F))$, for any behaviour tree t and any function $F : \text{seq}.\Sigma \rightarrow \text{behTree}.\Sigma$.

Example 2. Consider the sequential composition of programs S (originally shown in Fig. 3) and T in Fig. 6 from some initial state σ_0 . On the left of the diagram we show one behaviour tree t_S from $S.\sigma_0$, and the behaviour trees from T which may be used to extend each of its terminating branches. On the right we show the tree t which is created by function *concatTree*. We can see the probability of t to produce behaviour prefix $\langle 1 \rangle$ is one since: (a) t_S produces this prefix with probability $\frac{1}{2}$; and (b) one of its incomplete branches, $\langle - \rangle$, which is produced with probability $\frac{1}{2}$, is extended to produce this behaviour prefix. Since the tree has no miraculous branches, function *trim* would have no effect when applied to this tree.

Example 3. To gain a better understanding of how the function *trim* works, the reader may wish to consider the sequential composition of programs M and N

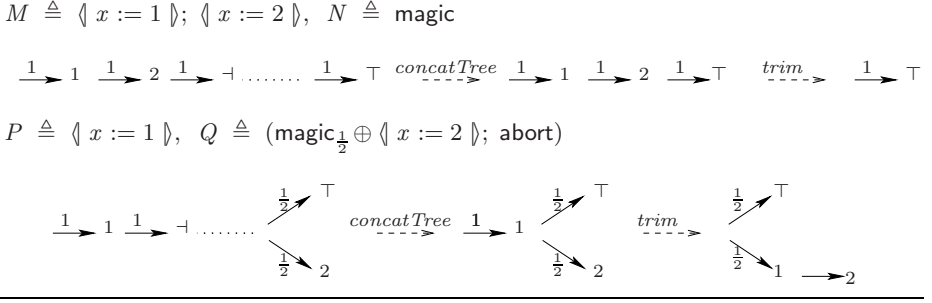


Fig. 7. Reactive programs M and N and the construction of one possible behaviour tree from $(M; N).\sigma_0$; and programs P and Q and the construction of a possible behaviour tree from $(P; Q).\sigma_0$, for some initial state σ_0 .

from Fig. 7. The first part of the figure shows how a behaviour tree from $M.\sigma_0$ may be concatenated together with a tree from N to produce a relaxed behaviour tree; and the second part demonstrates how *trim* acts on this tree to effectively remove the branch which has been marked as miraculous, replacing it with a miraculous behaviour at the root. This behaviour may seem odd, however it is important to treat miraculous behaviour in this way so that guards may be used to constrain the paths that are taken in a program. For example, we have that $(\langle x := 1 \rangle \sqcap \langle x := 2 \rangle); [x = 1] = \langle x := 1 \rangle$. A more complex example, in which probabilistic behaviour is also present, is given in Fig. 7.

Now that sequential composition has been defined, we can more clearly see the role that the distinguished programs *skip*, *magic*, *abort*, and hence guards and assertions play. *skip* is indeed the unit of sequential composition: for any reactive program S , $\text{skip}; S = S$; $\text{skip} = S$. Top program *magic* is a left, but not necessarily a right annihilator of sequential composition. When composed on the right it is only able to affect trees with terminating branches. The aborting and nonterminating branches of a tree are untouched. The least program *abort*, like *magic*, is a left, but not a right, annihilator. However, when composed on the right it is unable to constrain input trees, instead it simply extends incomplete branches by aborting.

Both weak, $*$, and strong, $^\omega$, iteration operators are defined using greatest and least fixpoints, respectively.⁶ T^* produces a program which performs T any finite number of times, while T^ω iterates T any finite or any infinite number of times. The strong iteration construct may be used to clearly illustrate the difference between the sequential probabilistic programs and the trace-based ones. For the non-reactive probabilistic model (Sect. 3.1), nonterminating behaviour is equated with program abortion, for instance $(x := 1)^\omega = \text{abort}$, however nonterminating behaviour in probabilistic reactive programs may produce infinite traces, as is

⁶ The existence of these fixpoints is guaranteed because our set of reactive programs forms a complete partial order with respect to $\sqsubseteq$.

the case for $\langle x := 1 \rangle^\omega$ or $\langle \text{skip} \rangle^\omega$. It is instructive to note, however, that skip^ω equals abort for both models.

Probabilistic Action Systems. A probabilistic action system [12] consists of an initialisation action, A_0 , and a set of probabilistic atomic actions $A_1, \dots, A_n$, which we express using probabilistic guarded commands. Each action A_i has an implicit guard $\text{gd}.A_i$ associated with it, $\text{gd}.A_i \triangleq (\lambda \sigma \in \Sigma \cdot A_i.\sigma \neq \{d_\top\})$, which identifies the states from which A_i is able to be executed. Each action is required to be feasible when its guard holds: that is we require $\{\text{gd}.A_i\}; A_i; \text{abort} = \text{abort}$.

The behaviour of such a probabilistic action system may be defined in the reactive language as $\langle A_0 \rangle; \text{do } \langle A_1 \rangle \sqcap \dots \sqcap \langle A_n \rangle \text{ od}$, where a do-loop $\text{do } S \text{ od}$, $\text{do } S \text{ od} \triangleq S^\omega; [\neg \text{gd}.S]$, iterates reactive program S until the guard of S , $\text{gd}.S \triangleq (\lambda \sigma \in \Sigma \cdot S.\sigma \neq \{t_\top\})$, ceases to hold.⁷

5 An Algebra of Reactive Programs

In this section we investigate the trace model for reactive probabilistic programs in the context of refinement algebra [17,18,14,8]. We show that the reactive programs form a general refinement algebra, in which guards and assertions may be defined abstract-algebraically, and that enabledness and termination operators have a natural interpretation in the trace model. We also briefly discuss how the action body operators and the trace operators interact, and applications of the algebra.

General Refinement Algebra. The general refinement algebra [18] is an abstract algebra which, as well as being suitable for total-correctness reasoning about non-reactive programs that may include both angelic and demonic choice, is appropriate for non-reactive probabilistic program models such as the one presented in Sect. 3.1 [8]. Although the general refinement algebra does not explicitly contain a probabilistic choice operator, it has been shown to be a very useful tool for reasoning about probabilistic program transformations: in recent work, many transformation theorems which may be expressed and verified in the general refinement algebra have been investigated [7].

Formally, a *general refinement algebra* is a structure over the signature $(\sqcap, ;, *, ^\omega, \top, 1)$ with ordering $x \sqsubseteq y \triangleq x \sqcap y = x$, such that the reduct over $(\sqcap, \top)$ is an idempotent, commutative monoid with identity element $\top$, the reduct over $(; , 1)$ is a monoid with identity element 1, sequential composition – which we leave implicit – distributes over demonic choice according to

$$x(y \sqcap z) \sqsubseteq xy \sqcap xz \text{ and } (x \sqcap y)z = xz \sqcap yz,$$

⁷ Note that probabilistic action systems *do not* exhibit miraculous behaviour and so they could also be defined using a model in which such behaviour is not expressible. (In fact Sere and Troubitsyna [12] express non-reactive probabilistic action systems in a model without magic.) As mentioned in Sect. 2 the problem with using a model without magic is that we would not be able to express the guards of actions as programs themselves. This would lead to a more complicated algebraic treatment.

$\top$ annihilates from the right: $\top x = \top$ – and the weak and the strong iteration operators, $*$ and $^\omega$, satisfy unfolding and induction axioms

$$\begin{aligned} x^* &= 1 \sqcap x x^* & \text{and} & & x \sqsubseteq y x \sqcap z \Rightarrow x \sqsubseteq y^* z, & \text{and} \\ x^\omega &= 1 \sqcap x x^\omega & \text{and} & & y x \sqcap z \sqsubseteq x \Rightarrow y^\omega z \sqsubseteq x, \end{aligned}$$

respectively. A syntactic constant $\perp$, which represents the least element, is defined in the algebra by 1^ω .

Reactive probabilistic programs with the trace semantics given above forms a general refinement algebra, as stated by the following proposition.

Proposition 1. *For any state space Σ , $(\text{React}_\Sigma, \sqcap, ;, *, ^\omega, \text{magic}, \text{skip})$ is a general refinement algebra, with least element **abort** [8].*

A *guard* of a general refinement algebra is an element g of the carrier set such that it

- is right distributive, $g(x \sqcap y) = gx \sqcap gy$, and
- has a right distributive complement $\bar{g}$ satisfying $\bar{g}g = g\bar{g} = \top$ and $g \sqcap \bar{g} = 1$.

Given a guard, the corresponding *assertion*, g° , is defined by $g^\circ \triangleq \bar{g} \perp \sqcap 1$. It can be shown that the model-theoretic guards and assertions (as defined in Fig. 4) satisfy the above properties.

Enabledness and Termination. The enabledness operator of refinement algebra allows action systems to be expressed with implicit guards (see [14] for a more thorough treatment). In this paper, the operator ϵ , which takes an element of the carrier set and returns a guard, is axiomatised by

$$\epsilon x x = x, \quad g \sqsubseteq \epsilon(gx) \quad \text{and} \quad \epsilon(xy) = \epsilon(x\epsilon y).$$

In our model, ϵS is given the trace-interpretation $[\text{gd}.S]$. It can be shown that this interpretation satisfies the above axioms. In some axiomatisations (such as that which appears in [13]), a fourth axiom $\epsilon x \perp = x \perp$ has been added to the three axioms above. This axiom does not hold in the trace model: one reason for this is that the least element **abort** does not destroy trace behaviour which has already been produced (if it were capable of this then we would have that $\langle \text{skip} \rangle^\omega = \text{abort}$). Consider reactive program $\langle \text{skip} \rangle$. We have that $[\text{gd}.\langle \text{skip} \rangle]; \text{abort} = \text{skip}; \text{abort} = \text{abort}$, which is not equal to $\langle \text{skip} \rangle; \text{abort}$. The invalidity of the fourth axiom in a simpler trace model was also observed in [25].

⁸ It has been shown that probabilistic program models in which all elements distribute over non-empty codirected sets (that is $y; (\sqcap_{x \in X} \bullet x) = (\sqcap_{x \in X} \bullet y; x)$, for each element y and non-empty, codirected set X) have an additional induction rule, $x \sqsubseteq x(y \sqcap 1) \sqcap z \Rightarrow x \sqsubseteq zy^*$ [7] (which is included in probabilistic Kleene algebra [6], which may be used for reasoning about non-reactive probabilistic programs in a *partial correctness* framework, in monodic tree Kleene algebra [15] and in probabilistic demonic refinement algebra [8]). This axiom would also hold here if our reactive probabilistic programs distribute over non-empty codirected sets.

The termination operator τ is the “dual” of the enabledness operator [14]. For any element of the carrier set x , τx is defined to be an assertion which satisfies the axioms

$$x = \tau x x, \tau(g^\circ x) \sqsubseteq g^\circ, \tau(x\tau y) = \tau(xy) \text{ and } \tau(x \sqcap y) = \tau x \sqcap \tau y.$$

and for reactive program S it is given a trace-interpretation $\{\text{term}.S\}$, where $\text{term}.S = (\lambda \sigma \in \Sigma \bullet S.\sigma \neq \text{behTree}.\Sigma)$, denotes the states from which S does not behave like **abort** with probability one. Similarly to enabledness, the termination operator is sometimes characterised by a fourth axiom $\tau x \top = x \top$. Analogously to enabledness, this axiom is invalid in the trace model. One reason being that the top reactive program **magic** is unable to constrain tree branches which do not terminate. Consider program $S = \langle \text{skip} \rangle; \text{abort}$, we have that $\{\text{term}.S\}; \text{magic} = \text{skip}; \text{magic} = \text{magic}$ which does not equal $\langle \text{skip} \rangle; \text{abort}; \text{magic} = \langle \text{skip} \rangle; \text{abort}$.

Other Algebraic Properties. How do the action-body operators and the trace operators interact? First we may observe that the choice operators may be used to split atomic actions. That is,

$$\langle A_p \oplus A' \rangle = \langle A \rangle_p \oplus \langle A' \rangle \text{ and } \langle A \sqcap A' \rangle = \langle A \rangle \sqcap \langle A' \rangle.$$

We also have that guards and assertions may be shifted outside atomic actions under some circumstances:

$$\langle [g]; A \rangle = [g]; \langle A \rangle, \langle A \rangle, \langle A; [g] \rangle = \langle A \rangle; [g] \text{ and } \langle \{g\}; A \rangle = \{g\}; \langle A \rangle.$$

Note that we cannot shift assertions to the right. Take for instance reactive program $\langle \text{abort} \rangle$. This is equivalent to $\langle \text{skip}; \text{abort} \rangle$, which is not the same as $\langle \text{skip} \rangle; \text{abort}$.

Naturally, sequential program refinement can be used to prove refinement between atomic actions. That is, for atomic actions $\langle A \rangle$ and $\langle B \rangle$, $\langle A \rangle \sqsubseteq \langle B \rangle \Leftrightarrow A \sqsubseteq B$. Finally, we note that $\langle A \rangle$ is enabled if and only if A is, and it certainly terminates if and only if A does, that is, $\text{gd}.\langle A \rangle = \text{gd}.A$ and $\text{term}.\langle A \rangle = \text{term}.A$ hold, where $\text{term}.A \triangleq (\lambda \sigma \in \Sigma \bullet A.\sigma \neq \Sigma_\top)$.

Reuse and Action Systems. The fact that reactive programs form a general refinement algebra lets us reuse all properties that have been derived in the algebra in previous work. This means that we do not have to re-prove algebraic properties directly in the model, but we can just collect old ones. For example, in general refinement algebra it is known that properties

$$x^\omega = x^\omega(x \sqcap 1) \sqcap 1, x(yx)^\omega \sqsubseteq (xy)^\omega x \text{ and } (x \sqcap y)^\omega = x^\omega(yx^\omega)^\omega$$

hold [18,8].

Action systems have been formulated using the enabledness operator and investigated in demonic refinement algebra (a slightly stronger algebra than the

general one) [14]. By inspecting the proofs for action-system leapfrog and decomposition [14], it can be seen that we can reuse the proofs to yield the following properties.

Firstly, we can give a leapfrog rule for probabilistic action systems (that is, action systems in which the actions are allowed to be probabilistic). For elements x and y in the carrier set, if for all guards p and q , $\top = px\bar{q} \Rightarrow pxq \sqsubseteq px$, then

$$x \text{ do } y \text{ od} \sqsubseteq \text{do } x \text{ y od } x,$$

holds, where an action system loop $\text{do } x_1 \sqcap \dots \sqcap x_n \text{ od}$ is encoded as $(x_1 \sqcap \dots \sqcap x_n)^{\omega \overline{\epsilon x_1} \dots \overline{\epsilon x_n}}$. Although the condition for this rule, $\top = px\bar{q} \Rightarrow pxq \sqsubseteq px$ may not be shown to hold in the algebra [18], it may easily be shown to hold for the reactive probabilistic programs we have defined.⁹

Secondly, we have a decomposition rule for probabilistic action systems. That is, for elements x and y in the carrier set, if $1 = \epsilon(\text{do } y \text{ od})$, then

$$\text{do } x \sqcap y \text{ od} = \text{do } y \text{ od}; \text{do } x; \text{do } y \text{ od} \text{ od}$$

holds. Note that in demonic refinement algebra, the assumptions made in the above rules could actually be proved and did not need to be stipulated [14].

Using the extra algebraic rules defined in the previous subsection, we see that the decomposition rule may be also applied to action systems of the form $\langle A_0 \rangle; \text{do } \langle A_1 \sqcap A_2 \rangle \text{ od}$, since $\langle A_1 \sqcap A_2 \rangle$ may be split into a choice between two reactive programs.

6 Concluding Remarks

We have presented a probabilistic reactive language with trace semantics which is capable of modelling probabilistic action systems. This language makes it possible for us to express and reason about transformations of probabilistic action systems, and other reactive probabilistic programs, in the same way it is possible to express transformations of non-reactive probabilistic programs using the probabilistic guarded command language. Trace-based models have been developed for other probabilistic reactive languages, notably Segala presented a trace-based semantics for probabilistic labelled transition systems [11]. Our trace model is similar to, but different from Segala's since our model is state, and not event based, and because we allow aborting and miraculous behaviours to be modelled. Also, unlike Segala, we define guards, assertions, sequential composition, and other operators in the trace model. This is a novel contribution since these operators and primitives have not been (to the best of our knowledge) defined using probabilistic trace semantics.

By demonstrating that our probabilistic trace model forms a general refinement algebra with enabledness and termination operators, we have shown that

⁹ Which means that the general refinement algebra is not complete for our reactive probabilistic programs, just as it is not complete for the non-reactive probabilistic program model that we use to model the action bodies.

it is possible to reuse many of the theorems that have been developed for non-reactive (probabilistic) programs.

For simplicity and brevity we have not specified a trace semantics in which it is possible to “hide” *local* state, or finite sequences of stuttering steps (steps which do not modify the *global* state). Such a semantics could be used to specify “typical” probabilistic action systems in which the state is divided into a local and global component and local state information is hidden. The specification of such a richer model (which we conjecture would also form a general refinement algebra) would open up more opportunities to reason about program transformations. We could, for instance, reuse our existing algebraic proofs of data refinement [7], to show that data refinements which only modify local state are valid for probabilistic action systems with trace semantics. In more future work, we would like to demonstrate how program transformations could be used to aid the development of reactive probabilistic systems.

Acknowledgements. The authors are grateful to Ian J. Hayes for valuable discussions, and the anonymous referees for their helpful suggestions.

References

1. Back, R.J.R., von Wright, J.: Trace refinement of action systems. In: Jonsson, B., Parrow, J. (eds.) CONCUR 1994. LNCS, vol. 836, pp. 367–384. Springer, Heidelberg (1994)
2. Desharnais, J., Möller, B., Struth, G.: Algebraic notions of termination. Technical Report 2006-23, Institute of Computer Science, University of Augsburg (2006)
3. Hayes, I.J., Utting, M.: A sequential real-time refinement calculus. *Acta Informatica* 37(6), 385–448 (2001)
4. Hoare, C.A.R., He, J.: *Unifying Theories of Programming*. Prentice Hall, Englewood Cliffs (1998)
5. Höfner, P., Struth, G.: Algebraic notions of non-termination. Technical Report CS-06-12, Department of Computer Science, University of Sheffield (2006)
6. McIver, A.K., Cohen, E., Morgan, C.C.: Using probabilistic Kleene algebra for protocol verification. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 296–310. Springer, Heidelberg (2006)
7. Meinicke, L.A., Hayes, I.J.: Reasoning algebraically about probabilistic loops. In: Liu, Z., He, J. (eds.) ICFEM 2006. LNCS, vol. 4260, pp. 380–399. Springer, Heidelberg (2006)
8. Meinicke, L.A., Solin, K.: Refinement algebra for probabilistic programs. In: Boiten, E., Derrick, J., Smith, G. (eds.) REFINE (to appear in ENTCS, 2007)
9. Möller, B.: Lazy Kleene algebra. In: Kozen, D. (ed.) MPC 2004. LNCS, vol. 3125, pp. 252–273. Springer, Heidelberg (2004)
10. Morgan, C.C., McIver, A.K.: Cost analysis of games, using program logic. In: APSEC 2001, p. 351. IEEE Computer Society Press, Washington, DC, USA (2001)
11. Segala, R.: A compositional trace-based semantics for probabilistic automata. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 234–248. Springer, Heidelberg (1995)
12. Sere, K., Troubitsyna, E.A.: Probabilities in action systems. In: Proc. of the 8th Nordic Workshop on Programming Theory (1996)

13. Solin, K.: On two dually nondeterministic refinement algebras. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 373–387. Springer, Heidelberg (2006)
14. Solin, K., von Wright, J.: Refinement algebra with operators for enabledness and termination. In: Uustalu, T. (ed.) MPC 2006. LNCS, vol. 4014, pp. 397–415. Springer, Heidelberg (2006)
15. Takai, T., Furusawa, H.: Monodic tree Kleene algebra. In: Schmidt, R.A. (ed.) RelMiCS/AKA 2006. LNCS, vol. 4136, pp. 402–416. Springer, Heidelberg (2006)
16. Troubitsyna, E.A.: Reliability assessment through probabilistic refinement. *Nordic Journal of Computing*, 320–342 (1999)
17. von Wright, J.: From Kleene algebra to refinement algebra. In: Boiten, E.A., Möller, B. (eds.) MPC 2002. LNCS, vol. 2386, pp. 233–262. Springer, Heidelberg (2002)
18. von Wright, J.: Towards a refinement algebra. *Science of Computer Programming* 51, 23–45 (2004)

Knowledge and Games in Modal Semirings

Bernhard Möller

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

moeller@informatik.uni-augsburg.de

Abstract. Algebraic logic compacts many small steps of general logical derivation into large steps of equational reasoning. We illustrate this by representing epistemic logic and game logic in modal semirings and modal Kleene algebras. For epistemics we treat the classical wise men puzzle and show how to handle knowledge update and revision algebraically. For games, we generalise the well-known connection between game logic and dynamic logic to modal semirings and link it to predicate transformer semantics, in particular to demonic refinement algebra. The study provides evidence that modal semirings will be able to handle a wide variety of (multi-)modal logics in a uniform algebraic fashion well suited to machine assistance.

1 Introduction

Algebraic logic strives to compact many small steps of general logical derivation into large steps of equational reasoning. On the semantic side, it attempts to replace tedious model-theoretic argumentation by more abstract reasoning.

A very useful algebraic structure for this are *semirings* (e.g. [18]) that abstract (state) transition systems by axiomatising their fundamental operations choice and sequential composition. Semirings with idempotent choice have a natural approximation order that corresponds to implication, so that implicational inference is replaced by inequational reasoning. Adding finite and infinite iteration leads to Kleene algebras [16] and omega algebras [7].

Modal semirings [9] are based on the concept of *tests* [17] that represent state predicates algebraically. They add diamond and box operators and are more general than Kripke structures, since the access between possible worlds need not be described by relations, but, e.g., by sets of computation paths or even by computation trees. Adding finite and infinite iteration yields modal Kleene and omega algebras which admit algebraic semantics of PDL, LTL and CTL; the subclass of left Boolean quantales can even handle full CTL* and the propositional μ -calculus [22]. Many further applications have been developed.

Here we show that modal semirings also lead to uniform and useful algebraisations of epistemic and game logics (e.g. [14,26]). For the former we treat the classical wise men puzzle and show how knowledge update and revision operators can be defined algebraically. For the latter we extend the well-known connection with PDL to the more general case of modal semirings and link it to predicate transformer semantics, in particular to demonic refinement algebra [28].

The framework is intended to be used for defining the semantics of new, special-purpose modal logics as they arise, e.g., with multi-agent systems. The advantage of using it is that many standard modal properties such as axioms M and K as well as certain induction rules hold automatically and don't need to be proved separately for each new logic.

The paper is organised as follows. Part I deals with an algebraisation of epistemic logic. This logic is recapitulated in Section 2 and illustrated with a variant of the Wise Men Puzzle. Section 3 defines modal (left) semirings and Kleene algebras and lists the most essential properties of the box and diamond operators. They are applied in Section 4 to represent the usual epistemic operators of multiagent systems algebraically. The laws these inherit from the general algebraic framework are used in Section 5 for a concise solution of the Wise Men Puzzle. Section 6 shows further use of the algebra in modelling certain aspects like preference relations between possible worlds and knowledge revision.

Part II treats games and predicate transformers. Section 7 provides a brief recapitulation of games and their algebra, in particular, of their representation as predicate transformers. These are analysed in a general fashion in Section 8, and a connection to Parikh's iteration operators for games is set up. Section 9 extends the left semiring of predicate transformers to a modal one and relates the box and diamond operators there to the enabledness and termination operators of demonic refinement algebra. Section 10 provides a brief conclusion and outlook.

Part I: Knowledge

We first model epistemic logic in modal semirings. As our running example we use a particular version of the Wise Men Puzzle [19].

2 The Wise Men Puzzle and Epistemic Modal Logic

A king wants to test the wisdom of his three wise men. They have to sit on three chairs behind each other, all facing the same direction. The king puts a hat on each head, either red or black, in such a way that no one can see his own hat, only the hats of the men before him. Then the king announces that at least one hat is red. He asks the wise man in the back if he knows his hat colour, but that one denies. Then he asks the middle one who denies, too. Finally he says to the front one: "If you are really wise, you should now know the colour of your hat."

To treat the puzzle in epistemic logic, one uses formulae $K_j\varphi$ (man j knows φ , *individual knowledge*), $E\varphi$ (*everyone knows* φ) or $C\varphi$ (everyone knows that everyone knows that ... that everyone knows φ , i.e., φ is *common knowledge*).

Let the men be numbered in the order of questioning, i.e., from back to front, and let r_i mean that i 's hat is red. Then we have the following assertions about common knowledge, since everyone hears what is being said:

- Every man can only see the hats before him, i.e., for $j < i$,
 $C(r_i \rightarrow K_j r_i)$ and $C(\neg r_i \rightarrow K_j \neg r_i)$.

- At least one hat is red, i.e., $C(r_1 \vee r_2 \vee r_3)$.
- After the king's questions, for $i = 1, 2$ we have $C(\neg K_i r_i)$ and $C(\neg K_i \neg r_i)$.

Can we infer anything about $K_3 r_3$ from that?

The aim of Part I is to give an algebraic semantics for the knowledge operators and to solve the puzzle by (in)equational reasoning.

To prepare the algebraisation we recall the main elements of Kripke semantics for modal logic (e.g. [14]). We will use a multiagent setting (each wise man is an agent) in which each agent has his own box and diamond operators.

A (multimodal) Kripke frame is a pair $K = (W, R)$, where W is a set of possible worlds and $R = (R_i)_{i \in I}$, for some index set I , is a family of binary access relations $R_i \subseteq W \times W$ between worlds.

The satisfaction relation $K, w \models \varphi$ tells whether a formula φ holds in world w in frame K . A formula characterises the subset $\llbracket \varphi \rrbracket =_{df} \{w \mid K, w \models \varphi\}$ of possible worlds in which it holds.

The semantics of the modal operators $\langle R_i \rangle$ and $[R_i]$ is given by

$$\begin{aligned} w \in \llbracket \langle R_i \rangle \varphi \rrbracket &\Leftrightarrow_{df} \exists v : R_i(w, v) \wedge v \in \llbracket \varphi \rrbracket, \\ w \in \llbracket [R_i] \varphi \rrbracket &\Leftrightarrow_{df} \forall v : R_i(w, v) \Rightarrow v \in \llbracket \varphi \rrbracket. \end{aligned}$$

In epistemic logic the worlds accessible from a current world w through R_i are called the *epistemic R_i -neighbours* of w . The knowledge of agent i in a world w consists of the formulae that are true in all epistemic neighbours of w (which in presence of axiom (T) below include w itself). Therefore, the knowledge operator K_i coincides with $[R_i]$, whereas its de Morgan dual $\langle R_i \rangle$ coincides with the possibility operator P_i .

Usually, special axioms for the knowledge operators are required:

$$\begin{array}{ll} K_i \varphi \rightarrow \varphi & \text{if } i \text{ knows } \varphi \text{ then } \varphi \text{ is actually true (truth)} \quad (T) \\ K_i \varphi \rightarrow K_i K_i \varphi & \text{if } i \text{ knows } \varphi, \text{ he knows that (positive introspection)} \quad (PI) \\ \neg K_i \varphi \rightarrow K_i \neg K_i \varphi & \text{analogous (negative introspection)} \quad (NI) \end{array}$$

We will see in the solution of the puzzle which of these are actually needed.

3 Algebraic Semantics: Modal Semirings

There are already various algebraisations of modal operators, e.g., Boolean algebras with operators [15] and propositional dynamic logic PDL [12]. Moreover, a partly algebraic treatment of Kripke frames can be given using relation algebra; the knowledge requirements above correspond to the following relational ones:

$$\begin{array}{lll} K_i \varphi \rightarrow \varphi & \Delta \subseteq R_i & \text{reflexivity} \\ K_i \varphi \rightarrow K_i K_i \varphi & R_i ; R_i \subseteq R_i & \text{transitivity} \\ \neg K_i \varphi \rightarrow K_i \neg K_i \varphi & R_i^\smile ; R_i \subseteq R_i & \text{euclidean property} \end{array}$$

Here, Δ is the diagonal or identity relation, $;$ is relational composition and $^\smile$ is relational converse.

Modal semirings and Kleene algebras provide a very effective combination of PDL and algebraic operations on the access relations. Additionally, they abstract

from the special case of access *relations* and allow more general access elements such as sets of computation paths. The particular subclass of Boolean quantales allows the incorporation of infinite iteration and μ -calculus-like recursive definitions, rendering it suitable for handling even full CTL* [22].

A *left semiring* is a structure $(S, +, 0, \cdot, 1)$ with axioms to be detailed below. In most applications these operators are interpreted as follows:

$$\begin{aligned} + &\leftrightarrow \text{choice,} & \cdot &\leftrightarrow \text{sequential composition,} \\ 0 &\leftrightarrow \text{empty choice,} & 1 &\leftrightarrow \text{null action,} \\ \leq &\leftrightarrow \text{increase in information or in choice possibilities.} \end{aligned}$$

The axioms of a left semiring are now as follows.

- The reduct $(S, +, 0)$ is a commutative and idempotent monoid. This induces the *natural order* $a \leq b \Leftrightarrow_{df} a + b = b$ w.r.t. which 0 is the least element and $a + b$ is the join of a and b .
- The reduct $(S, \cdot, 1)$ is a monoid.
- Composition $\cdot$ is left-distributive and left-strict, i.e., $(a + b) \cdot c = a \cdot c + b \cdot c$ and $0 \cdot a = 0$.
- Composition is $\leq$ -isotone in its right argument, i.e., $b \leq c \Rightarrow a \cdot b \leq a \cdot c$.

A *weak semiring* is a left semiring in which composition is also right-distributive. A weak semiring with right-strictness is called a *full semiring* or simply *semiring*. All these requirements can be axiomatised purely equationally.

A prominent full semiring is the set of all binary relations over a set W with union as $+$ and relational composition as $\cdot$.

A proper left semiring structure is at the core of process algebra frameworks (e.g. [6]); for further discussion of the connections see [21].

While general semiring elements can be thought of as sets of transitions or transition paths between states, we now describe how to model state predicates or, isomorphically, sets of states algebraically by tests. A *test* is a subidentity $p \leq 1$ that has a complement $\neg p$ relative to 1, i.e., $p \cdot \neg p = 0 = \neg p \cdot p$ and $p + \neg p = 1$. If p characterises a set S of states then $\neg p$ characterises its complement. Note that $\neg$ is required only for tests, not for general semiring elements, which allows a much wider class of models. The set of all tests of S is denoted by $\text{test}(S)$.

In the relation semiring, the tests are the subidentities of the form $\Delta_V =_{df} \{(x, x) \mid x \in V\}$ for subsets $V \subseteq W$. So Δ_V can represent V as a relation and hence model the predicate characterising V .

The above definition of tests deviates slightly from that in [17] in that it does not allow an arbitrary Boolean algebra of subidentities as $\text{test}(S)$ but only the maximal complemented one. The reason is that the axiomatisation of box to be presented below forces this maximality anyway (see [8]).

Straightforward calculations show that $\text{test}(S)$ forms a Boolean algebra with $+$ as join, $\cdot$ as meet and 0 and 1 as its least and greatest elements. We will consistently write $a, b, c, \dots$ for arbitrary semiring elements and $p, q, r, \dots$ for

tests. When tests are viewed as predicates over a set W of possible worlds, the semiring operators play the following roles:

$$\begin{aligned}
0 / 1 &\leftrightarrow \text{false (empty set) / true (full set } W), \\
+ / \cdot &\leftrightarrow \text{disjunction (union) / conjunction (intersection),} \\
\leq &\leftrightarrow \text{implication (subsethood),} \\
p \cdot a / a \cdot p &\leftrightarrow \text{input / output restriction of } a \text{ by } p, \\
p \cdot a \cdot q &\leftrightarrow \text{the part of } a \text{ taking } p\text{-elements to } q\text{-elements.} \quad (*)
\end{aligned}$$

To ease reading, we will write $\wedge$ and $\vee$ instead of $\cdot$ and $+$ when both of their arguments are tests (metalogical conjunction and disjunction will be denoted with the larger $\wedge$ and $\vee$ to avoid confusion). Also, we will freely use the standard Boolean operations on $\text{test}(S)$, for instance implication $p \rightarrow q =_{df} \neg p \vee q$ and relative complementation $p - q =_{df} p \wedge \neg q$, with their usual laws, notably the Galois connection (*shunting rule*) $p \wedge q \leq r \Leftrightarrow p \leq q \rightarrow r$ with the special case $q \leq r \Leftrightarrow 1 \leq q \rightarrow r$.

We now axiomatise a box operator $[-] : S \rightarrow (\text{test}(S) \rightarrow \text{test}(S))$. For semiring element a and test q the test $[a]q$ characterises those states for which all successor states under a satisfy q ; this coincides with the classical semantics of $[-]$ in multimodal logics (see e.g. [14]). The axioms are [23]

$$p \leq [a]q \Leftrightarrow p \cdot a \cdot \neg q = 0, \quad (b1) \qquad [a \cdot b]p = [a][b]p. \quad (b2)$$

According to $(*)$ above, axiom (b1) means that all p -worlds satisfy $[a]q$ iff there is no a -connection from p -worlds to $\neg q$ -worlds. This specifies $[a]q$ as the weakest of all such predicates; box is the abstract counterpart of the weakest liberal precondition predicate transformer wlp [11], with $p \leq [a]q$ representing the partial correctness semantics of the Hoare triple $\{p\} a \{q\}$. Axiom (b2) makes box well-behaved w.r.t. composition. Diamond is the de Morgan dual of box and by (b2) is again well-behaved w.r.t. composition:

$$\langle a \cdot b \rangle p = \langle a \rangle \langle b \rangle p, \qquad \langle a \rangle p =_{df} \neg[a] \neg p \quad (1)$$

A (left/weak) semiring with box (and hence diamond) is called *modal*. Both operators are unique if they exist. They coincide with the corresponding ones in PDL (e.g. [14]); the difference is that in PDL the first arguments a of the box are of a purely syntactic nature without any algebraic laws.

An equivalent purely equational axiomatisation via a domain operator has been presented in [8] for the case of a full semiring. In [21] it has been shown that it carries over to left semirings.

We list some useful properties. De Morgan duality gives the swapping rule

$$\langle a \rangle [b] p \leq [c] p \Leftrightarrow \langle c \rangle p \leq [a] \langle b \rangle p. \quad (2)$$

Box is anti-disjunctive and diamond is disjunctive in the first argument:

$$[a + b] p = [a] p \wedge [b] p, \qquad \langle a + b \rangle p = \langle a \rangle p \vee \langle b \rangle p. \quad (3)$$

Hence box is antitone and diamond is isotone in the first argument: if $a \leq b$ then

$$[a] p \geq [b] p, \qquad \langle a \rangle p \leq \langle b \rangle p.$$

To understand the antitony, recall that the implication order $a \leq b$ expresses that b offers at least as many transition possibilities as a . Now, if more choices

are offered, one can guarantee less, which is expressed by $[b]p \leq [a]p$. Finally, for tests box and diamond can be given explicitly:

$$[p]q = p \rightarrow q, \quad \langle p \rangle q = p \wedge q. \quad (4)$$

This agrees with the behaviour of the test operation $p?$ in PDL.

Next, we describe finite iteration. A (left/weak) Kleene algebra [16] is a structure $(S, +, 0, \cdot, 1, *)$ such that the reduct $(S, +, 0, \cdot, 1)$ is a (left/weak) semiring and the finite iteration operator $*$ satisfies the left unfold and induction axioms

$$1 + a \cdot a^* \leq a^*, \quad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c.$$

In the relation semiring, a^* and $a^+ =_{df} a^* \cdot a$ are the reflexive-transitive and transitive closure of a , respectively.

A (left/weak) Kleene algebra is *modal* when the underlying left/weak semiring is. In this case the axioms entail box and diamond star and plus induction [8]:

$$q \leq p \wedge [a]q \Rightarrow q \leq [a^*]p, \quad p \vee \langle a \rangle q \leq q \Rightarrow \langle a^* \rangle p \leq q \quad (5)$$

$$q \leq [a]p \wedge [a]q \Rightarrow q \leq [a^+]p, \quad \langle a \rangle p \vee \langle a \rangle q \leq q \Rightarrow \langle a^+ \rangle p \leq q. \quad (6)$$

Using Hoare triples the box part of (5) reads $(q \Rightarrow p \wedge \{q\} a \{q\}) \Rightarrow \{q\} a^* \{p\}$, which is related to the familiar Hoare rule for the while loop. Moreover, we have the PDL induction rules (see [23])

$$[a^*](p \rightarrow [a]p) \leq p \rightarrow [a^*]p, \quad \langle a^* \rangle - 1 \leq \langle a^* \rangle (\langle a \rangle - 1). \quad (7)$$

4 Knowledge Algebra

Using our modal operators we can now model common knowledge over a left semiring S as follows. Assume a finite set of agents, represented by an index set $I = \{1, \dots, n\}$, each with an accessibility element $a_i \in S$. An *agent group* is a subset $G \subseteq I$. We introduce two operators for expressing common knowledge:

- $E_G p$: everyone in group G knows p
- $C_G p$: everyone in G knows that everyone in G knows that ... that p holds.

Using antisdisjunctivity (3) of box we calculate, for $G = \{k_1, \dots, k_m\}$,

$$\begin{aligned} E_G p &= K_{k_1} p \wedge \dots \wedge K_{k_m} p = [a_{k_1}]p \wedge \dots \wedge [a_{k_m}]p \\ &= [a_{k_1} + \dots + a_{k_m}]p = [a_G]p, \end{aligned}$$

where $a_G =_{df} a_{k_1} + \dots + a_{k_m}$.

Likewise, using the composition axiom (b2) and again antisdisjunctivity (3) of box, we obtain, semi-formally,¹

$$\begin{aligned} C_G p &= E_G p \wedge E_G E_G p \wedge E_G E_G E_G p \wedge \dots \\ &= [a_G]p \wedge [a_G][a_G]p \wedge [a_G][a_G][a_G]p \wedge \dots \\ &= [a_G]p \wedge [a_G \cdot a_G]p \wedge [a_G \cdot a_G \cdot a_G]p \wedge \dots \\ &= [a_G + a_G^2 + a_G^3 \dots]p. \end{aligned}$$

¹ This notation is semi-formal, since general infinite products and sums need not exist in every left semiring; even if this particular one exists, it need not coincide with a_G^+ .

Therefore we define $C_G p =_{df} [a_G^+]p$ if the underlying semiring is a Kleene algebra.

In this way we have obtained an algebraic counterpart of the multiagent logic KT45ⁿ (e.g. [14]) and dynamic epistemic logic [3].

From antitony of box in its first argument we get, since $a_{k_j} \leq a_G \leq a_G^+$,

$$C_G p \leq E_G p \leq K_{k_j} p \qquad C_G p \leq C_G K_{k_j} p . \quad (8)$$

All our properties up to here hold irrespective of the knowledge axioms. Let us see what can be derived if these are assumed.

If all K_i are reflexive (i.e., satisfy axiom (T)) then so is E_G and hence $C_G = [a_G^*]$. Therefore the general induction rule (7) specialises to the knowledge induction rule

$$C_G(p \rightarrow E_G p) \leq p \rightarrow C_G p .$$

It means that if all agents in G know invariance of p under E_G and p is true then all agents know they all know p . Moreover, (b2) and a star property yield

$$C_G C_G p = [a_G^*][a_G^*]p = [a_G^* \cdot a_G^*]p = [a_G^*]p = C_G p$$

and hence, by conjunctivity of C_G ,

$$C_G p \wedge C_G q = C_G C_G p \wedge C_G C_G q = C_G(C_G p \wedge C_G q) . \quad (9)$$

As another application of the algebra we show that negative introspection is preserved under transitive closure (for positive introspection this is trivial, since that property is equivalent to transitivity, so that transitive closure does not add anything). To this end we use the equivalent formulations

$$NI(a) \Leftrightarrow_{df} \forall p . \langle a \rangle[a]p \leq [a]p \Leftrightarrow \forall p . \langle a \rangle p \leq [a]\langle a \rangle p$$

of that property to ease use of the above-mentioned (co-)induction rules.

Lemma 4.1. $NI(a) \Rightarrow NI(a^+)$.

Proof. The claim $\langle a^+ \rangle[a^+]p \leq \langle a^+ \rangle p$ reduces by the star induction axiom to $\langle a \rangle[a^+]p \vee \langle a \rangle \langle a^+ \rangle p \leq \langle a^+ \rangle p$, which splits into $\langle a \rangle[a^+]p \leq \langle a^+ \rangle p \wedge \langle a \rangle \langle a^+ \rangle p \leq \langle a^+ \rangle p$. The second conjunct follows by $\langle a \rangle \langle a^+ \rangle p = \langle a \cdot a^+ \rangle p$ and $a \cdot a^+ \leq a^+$ by isotony of $\langle \rangle$. For the first conjunct we calculate

$$\begin{aligned} & \langle a \rangle[a^+]p \leq \langle a^+ \rangle p \\ \Leftrightarrow & \langle a^+ \rangle p \leq [a]\langle a^+ \rangle p && \text{swapping rule (2)} \\ \Leftarrow & \langle a \rangle p \vee \langle a \rangle[a]\langle a^+ \rangle p \leq [a]\langle a^+ \rangle p && \text{induction (6)} \\ \Leftrightarrow & \langle a \rangle p \leq [a]\langle a^+ \rangle p \wedge \langle a \rangle[a]\langle a^+ \rangle p \leq [a]\langle a^+ \rangle p \end{aligned}$$

The second of these conjuncts holds by $NI(a)$. For the first one we continue, using the definition of a^+ and the composition rule (11),

$$\langle a \rangle p \leq [a]\langle a^+ \rangle p \Leftrightarrow \langle a \rangle p \leq [a]\langle a \rangle \langle a^* \rangle p \Leftarrow NI(a) \wedge p \leq \langle a^* \rangle p ,$$

and are done², since the second conjunct follows from $1 \leq a^*$ and $\langle 1 \rangle p = p$. $\square$

² The proof could be compacted even more by using a point-free style; e.g., $NI(a)$ is equivalent to $\langle a \rangle \circ [a] \leq \langle a \rangle$ where $\leq$ is now the pointwise lifting of the semiring order to predicate transformers.

5 Solving the Wise Men Puzzle

For the results of the present section we assume the underlying left semiring S to be weak. Then we have the following additional properties:

- Box is conjunctive and diamond is disjunctive:

$$[a](p \wedge q) = [a]p \wedge [a]q, \quad \langle a \rangle(p \vee q) = \langle a \rangle p \vee \langle a \rangle q.$$

- Hence both operators are isotone in the second argument: if $p \leq q$ then

$$[a]p \leq [a]q, \quad \langle a \rangle p \leq \langle a \rangle q.$$

- Moreover, Box satisfies axiom K of modal logic and diamond its dual:

$$[a](p \rightarrow q) \leq [a]p \rightarrow [a]q, \quad \langle a \rangle p - \langle a \rangle q \leq \langle a \rangle(p - q). \quad (\text{K})$$

By contraposition and shunting, this is equivalent to the following forms (modal modus tollens, given only for box):

$$[a](p \rightarrow q) \wedge \neg[a]q \leq \neg[a]p, \quad [a](p \vee q) \wedge \neg[a]q \leq \neg[a]\neg p. \quad (\text{K}')$$

- If S is full then box satisfies axiom M of modal logic and diamond its dual:

$$[a]1 = 1, \quad \langle a \rangle 0 = 0. \quad (\text{M})$$

Let us now use the algebra to solve the Wise Men Puzzle over a full semiring. First we define validity of a test p by $\models p \Leftrightarrow_{df} 1 \leq p$. By shunting, $\models q \rightarrow r \Leftrightarrow q \leq r$. Moreover, $\models p \wedge p \leq q \Rightarrow \models q$.

With this notation we can repeat the assumptions about the puzzle from Section 2 in a more precise form (the indices of C and E are suppressed, since always the full group of all three agents is referred to):

$$\begin{array}{lll} \text{(a)} \models C(r_i \rightarrow K_j r_i) & \text{(b)} \models C(\neg r_i \rightarrow K_j \neg r_i) & (j < i) \\ \text{(c)} \models C(r_1 \vee r_2 \vee r_3) & & \\ \text{(d)} \models C(\neg K_i r_i) & \text{(e)} \models C(\neg K_i \neg r_i) & (i = 1, 2) \end{array}$$

Our main reasoning principle is isotony: If f is an isotone function from tests to tests then $p \leq q \wedge \models f(p) \Rightarrow \models f(q)$. Since we have defined E and C as boxes, this principle applies to them without the need for a separate proof.

Now we assume that all K_i and hence E and C are reflexive. Starting from a conjunction of formulae of type (c) and (d), we reason as follows,

$$\begin{array}{ll} C(r_1 \vee r_2 \vee r_3) \wedge C(\neg K_1 r_1) & \\ = C(C(r_1 \vee r_2 \vee r_3) \wedge C(\neg K_1 r_1)) & \text{by (9)} \\ \leq C(K_1(r_1 \vee r_2 \vee r_3) \wedge \neg K_1 r_1) & \text{common knowledge (8) and reflexivity of C} \\ \leq C(\neg K_1 \neg(r_2 \vee r_3)) & \text{by (K')} \\ = C(\neg K_1(\neg r_2 \wedge \neg r_3)) & \text{de Morgan} \\ = C(\neg(K_1 \neg r_2 \wedge K_1 \neg r_3)) & \text{conjunctivity of } K_1 \\ = C(\neg K_1 \neg r_2 \vee \neg K_1 \neg r_3) & \text{de Morgan} \\ \leq C(r_2 \vee r_3) & \text{contrapositives of formulae (b)} \\ & \text{and reflexivity of C} \end{array}$$

Analogous reasoning shows $C(r_2 \vee r_3) \wedge C(\neg K_2 r_2) \leq C(r_3) \leq K_3(r_3)$ and we are done, since this means that the third wise man knows his hat is red, which by reflexivity (T) is indeed true.

This latter step also shows that the solution easily generalises to n instead of three wise men. In fact, one can give a closed form of the generalised argument: for an agent group G and a subgroup $H \subseteq G$ of agents who have already been interrogated and have denied knowledge of their hat colour,

$$C(\bigvee_{j \in G} r_j) \wedge C(\bigwedge_{i \in H} \neg K_i r_i) \wedge C(\bigwedge_{i \in H} \bigwedge_{j \in G-H} r_j \rightarrow K_i r_j) \leq C(\bigvee_{j \in G-H} r_j) .$$

Note that we have only used reflexivity of the knowledge modalities in Section 2, but neither positive nor negative introspection.

This argument can be re-used for puzzles with a similar structure, like the unexpected hanging paradox [29] or the muddy children [14], which adds several rounds of interrogation of the above shape. This works, because these puzzles have a “purely logical” structure. Contrarily, the puzzle about Mr. S and Mr. P [19] involves a lot of domain knowledge about arithmetic in addition to mutual knowledge of the agents about each other; therefore the abstract algebraic reasoning will cover only the overall structure of the solution, whereas the arithmetic details will take place within the test set of a particular semiring.

6 Preferences and Their Upgrade

We now return to our general setting of modal semirings; in particular we assume neither of the axioms (T), (PI) or (NI). Let us briefly show how one can reason about other aspects of knowledge and belief. Some agent logics allow expressing preferences between possible worlds (e.g. [5]).

Since we are completely free in choosing our accessibility elements, we can also include these. To this end we equip each agent i with his own preference relation $\preceq_i$. The intention is that $[\preceq_i]p$ holds in a world w iff p holds in all worlds that agent i prefers over w under $\preceq_i$.

Usually one requires that $\preceq_i$ be a preorder, modally expressed by

$$[\preceq_i]p \leq p , \quad [\preceq_i]p \leq [\preceq_i][\preceq_i]p .$$

Antisymmetry is not required: if $w_1 \preceq_i w_2 \wedge w_2 \preceq_i w_1$ then agent i is *indifferent* about w_1 and w_2 .

Using the preference concept, one can, e.g., model *regret* [5]: the formula $K_i \neg p \wedge \langle \preceq_i \rangle p$ expresses that although agent i knows that p is not true, he would still prefer a world where it would be.

A preference agent system can be updated in various ways. In *belief revision* agents may discard or add links to epistemic neighbour worlds. We model the two possibilities presented in [5] in our agent algebra.

In a *public announcement* of property p , denoted $!p$, one makes sure that all agents now know p . To this end, all links between p and $\neg p$ worlds are removed. In [5] this operator is explained in two ways:

- Satisfaction of $[!p]q$ in a frame is defined as satisfaction of q in a modified frame.

- The semantics is again given in a PDL-like fashion, making the new accessibility relation explicit in the first argument of box.

We can represent the latter approach directly in our setting by defining the modification of access element a_i as $a_i!p =_{df} p \cdot a_i \cdot p + \neg p \cdot a_i \cdot \neg p$. The advantage is that we now can just use the same algebraic laws as before and do not need to invent special inference rules for this operator.

Another change operation is *preference upgrade* by *suggesting* that p be observed. This affects the preference relations, not the accessibilities:

$$p\#\preceq_i =_{df} p \cdot \preceq_i \cdot p + \neg p \cdot \preceq_i \cdot \neg p.$$

Now agent i no longer prefers $\neg p$ worlds over p ones.

In the literature there are many more logics dealing with knowledge or belief revision. We are convinced that a large portion of these can be treated uniformly in the setting of modal semirings; for a related approach see [27], where belief update is modelled using semiring concepts.

Part II: Games and Predicate Transformers

In this part we return to the case of general left semirings.

7 Games and Their Algebra

The algebraic description of two-player games dates back at least to [25]; for a more recent survey see [26]. The idea is to use a predicate transformer semantics that is variant of (a μ -calculus-like enrichment of) PDL.

The starting point is, however, a slightly different relational model. It does not use relations of type $\mathcal{P}(W \times W)$, where the set of worlds W consists of the game positions and $\mathcal{P}$ is the power set operator, but rather of type $\mathcal{P}(W \times \mathcal{P}(W))$. A pair (s, X) in Relation R models that the player whose turn it is has a strategy to move from starting position s into a position in set X . To make this well-defined, R has to be $\subseteq$ -isotone in its second argument:

$$(s, X) \in R \wedge X \subseteq Y \Rightarrow (s, Y) \in R.$$

Now again, sets of worlds are identified with predicates over worlds. As pointed out in [25], such a relation R induces an isotone predicate transformer $\rho(R) : \mathcal{P}(W) \rightarrow \mathcal{P}(W)$ via $\rho(R)(X) =_{df} \{s \mid (s, X) \in R\}$. It is easy to check that the set of $\subseteq$ -isotone relations is isomorphic to that of isotone predicate transformers (both ordered by relational inclusion).

The basic operations to build up more complex games from atomic ones (such as single moves) are choice, sequential composition, finite iteration and tests, which are also basic operations found in left semirings; also the axioms (see [26]) are exactly those for left semirings. There are no constants 0 and 1; but they could easily be added by the standard extension of semigroups to monoids. The

only operation particular to game construction is *dualisation* in which the two players exchange their roles.

As games can be viewed as isotone predicate transformers, we study these from a bit more abstract viewpoint in the next section. Based on that we will show that they form a modal left semiring with dualisation, i.e., an abstract algebraic model of games. We will also show how to add finite iteration.

8 Predicate Transformers

For our purposes, all that matters about $\mathcal{P}(W)$ is its structure as a Boolean algebra. Therefore, more abstractly, a *predicate transformer* is a function $f : B \rightarrow B$, where B is an arbitrary Boolean algebra. As in Section 3 we denote the infimum, supremum and complementation operators by $\wedge$, $\vee$ and $\neg$, the least element by 0 and the greatest one by 1. Using $\vee$ for $+$ and $\wedge$ for $\cdot$ makes B a full modal semiring with $\text{test}(B) = B$ and $\langle p \rangle q = p \wedge q$ by (4).

If $p, q \in B$ and $f : B \rightarrow B$ satisfies $p \leq q \Rightarrow f(p) \leq f(q)$ then f is *isotone*. It is *disjunctive* if $f(p \vee q) = f(p) \vee f(q)$ and *conjunctive* if $f(p \wedge q) = f(p) \wedge f(q)$. It is *strict* if $f(0) = 0$ and *co-strict* if $f(1) = 1$. Finally, *id* is the identity transformer and $\circ$ denotes function composition.

Let $\text{PT}(B)$, $\text{ISO}(B)$, $\text{CON}(B)$ and $\text{DIS}(B)$ be the set of all, of isotone, of conjunctive and of disjunctive predicate transformers over B . It is well known that conjunctivity and disjunctivity imply isotony. Under the pointwise ordering $f \leq g \Leftrightarrow_{df} \forall p. f(p) \leq g(p)$, PT forms a lattice where the supremum $f \vee g$ and infimum $f \wedge g$ of f and g are the pointwise liftings of $\vee$ and $\wedge$, respectively:

$$(f \vee g)(p) =_{df} f(p) \vee g(p) , \quad (f \wedge g)(p) =_{df} f(p) \wedge g(p) .$$

The least and greatest elements of $\text{PT}(B)$ (and $\text{ISO}(B)$ and $\text{DIS}(B)$) are the constant functions $\mathbf{0}(p) =_{df} 0$ and $\mathbf{1}(p) =_{df} 1$. Note that $\mathbf{0}$ and $\mathbf{1}$ both are left zeros w.r.t. $\circ$. The substructure $(\text{ISO}, \vee, \mathbf{0}, \circ, id)$ is a left semiring; the substructure $(\text{DIS}(B), \vee, \mathbf{0}, \circ, id)$ is even a weak semiring. Likewise, the structure $(\text{CON}(B), \wedge, \mathbf{1}, \circ, id)$ is a weak semiring isomorphic to $\text{DIS}(B)$, but with the mirror ordering. The isomorphism is provided by the *duality operator* $^d : \text{PT}(B) \rightarrow \text{PT}(B)$, defined by $f^d(p) =_{df} \neg f(\neg p)$.

If $B = \text{test}(S)$ for some weak semiring S then the modal operator $\langle \_ \rangle$ provides a weak semiring homomorphism from S into $\text{DIS}(B)$.

If B is a complete Boolean algebra then $\text{PT}(B)$ is a complete lattice with $\text{ISO}(B)$, $\text{DIS}(B)$ and $\text{CON}(B)$ as complete sublattices. Hence we can extend $\text{ISO}(B)$ and $\text{DIS}(B)$ by a star operator via a least fixpoint definition:

$$f^* =_{df} \mu(\lambda g. id \vee f \circ g) ,$$

where μ is the least-fixpoint operator. It has been shown in [21] that this satisfies the star laws. By passing to the mirror ordering, one sees that also the subalgebra of conjunctive predicate transformers can be made into a left Kleene algebra; this is essentially the approach taken in [28] (except for infinite iteration).

A useful consequence of the star induction rule is a corresponding one for the dual of a star, generalising (5):

$$h \leq g \wedge f^d \circ h \Rightarrow h \leq (f^*)^d \circ g . \quad (10)$$

Let us now connect this to game algebra. For a predicate transformer g we find in [25] the following two definitions concerning iterations (we use boldface stars and brackets here to distinguish Parikh's notation from ours):

$$(a) \langle g^* \rangle p =_{df} \mu(\lambda y . p \vee g(y)) , \quad (b) [g^*]p =_{df} \nu(\lambda y . p \wedge g(y)) , \quad (11)$$

where ν is the greatest-fixpoint operator. Hence $\langle g^* \rangle$ in Parikh's notation coincides with g^* in ours. The defining functions of $\langle g^* \rangle$ and $[g^*]$ are de Morgan duals of each other; hence we can use the standard law $\nu f = \neg \mu f^d$ to calculate

$$\begin{aligned} [g^*](p) &= \nu(\lambda y . p \wedge g(y)) && \text{definition (11(b))} \\ &= \neg \mu(\lambda y . p \wedge g(y))^d && \text{above fixpoint law} \\ &= \neg \mu(\lambda y . \neg(p \wedge g(\neg y))) && \text{definition dual} \\ &= \neg \mu(\lambda y . \neg p \vee \neg g(\neg y)) && \text{de Morgan} \\ &= \neg \mu(\lambda y . \neg p \vee g^d(y)) && \text{definition dual} \\ &= \neg (g^d)^*(\neg p) && \text{definition (11(a))} \\ &= ((g^d)^*)^d(p) . && \text{definition dual} \end{aligned}$$

Thus, $[g^*]$ coincides with $((g^d)^*)^d$. This shows that we can fully represent game algebra with finite iteration in modal left Kleene algebras; the standard star axioms for iteration suffice. If desired, one could also axiomatise the dual of the star using the dualised unfold axiom $(f^*)^d \leq 1 \wedge f^d \circ (f^*)^d$ and (10) as the induction axiom.

Let us finally set up the connection with termination analysis. In [25] Parikh states that for concrete access relation R the predicate $\langle [R]^* \rangle \text{false}$ characterises the worlds from which no infinite access paths emanate. Plugging in the definitions for a general access element a we obtain

$$\langle [a]^* \rangle 0 = \mu(\lambda y . [a]y) .$$

This coincides with the *halting predicate* of the propositional μ -calculus [12]; in the semiring setting it and its complement have been termed the *convergence* and *divergence* of a and used extensively in [10]. They need not exist in arbitrary modal left semirings; rather they have to be axiomatised by the standard unfold and induction/co-induction laws for least and greatest fixpoints.

9 Modal Semirings of Predicate Transformers and Demonic Refinement Algebra

Although we have now seen a somewhat more abstract predicate transformer model of game algebra, we will now take one step further and present a modal left Kleene algebra of isotone predicate transformers. This will link game semantics directly with refinement algebra.

First we characterise the tests in the set $\text{ISO}(B)$; the proof of the following lemma can be found in the Appendix.

Lemma 9.1

1. $f \in \text{test}(\text{ISO}(B)) \Leftrightarrow f(p) = p \wedge f(1)$.
2. If $B = \text{test}(S)$ for some left semiring S then $\text{test}(\text{ISO}(B)) = \{\langle p \rangle \mid p \in B\}$.

Part 2. means that the tests in the semiring of isotone predicate transformers are precisely the diamonds of the elements of B (see Section 8).

Because of Part 1. and (4) we will, for convenience, denote mappings of the form $\lambda q. p \wedge q$ by $\langle p \rangle$ also in the general case of $\text{ISO}(B)$. The proof shows also that $\neg\langle p \rangle = \langle \neg p \rangle$.

Now we are ready to enrich $\text{ISO}(B)$ by box and diamond operators. To this end we work out what the right hand side of box axiom (b1) means there:

$$\begin{aligned} \langle p \rangle \circ f \circ \neg\langle q \rangle \leq 0 &\Leftrightarrow \forall r : p \wedge f(\neg q \wedge r) \leq 0 \Leftrightarrow p \wedge f(\neg q \wedge 1) \leq 0 \\ &\Leftrightarrow p \leq \neg f(\neg q) \Leftrightarrow p \leq f^d(q) ; \end{aligned}$$

the second equivalence holds by isotony of f . So the only possible choice is

$$[f]\langle q \rangle =_{df} \langle f^d(q) \rangle , \quad \langle f \rangle \langle q \rangle =_{df} \langle f(q) \rangle .$$

Let us check that this satisfies the second box axiom (b2) as well:

$$\begin{aligned} [f \circ g]\langle q \rangle &= \langle (f \circ g)^d(q) \rangle = \langle (f^d \circ g)^d(q) \rangle \\ &= \langle (f^d(g^d(q))) \rangle = [f]\langle g^d(q) \rangle = [f][g]\langle q \rangle . \end{aligned}$$

Hence box and diamond are well defined in $\text{ISO}(B)$. In sum:

Theorem 9.2. $\text{ISO}(B)$ forms a modal left Kleene algebra with dualisation.

This rounds off the picture in that now also the test operations of game algebra and PDL have become first-class citizens in predicate transformer algebra. Moreover, we can enrich that algebra by a domain operator which will provide the announced connection to refinement algebra.

Generally, in a modal left semiring the *domain operator* [8] $\ulcorner : S \rightarrow \text{test}(S)$ is given by $\ulcorner a =_{df} \langle a \rangle 1$. This characterises the set of starting worlds of access element a . For $\text{ISO}(B)$ this works out to $\ulcorner f = \langle f(1) \rangle$. This expression coincides with that for the termination operator τf in the concrete model of *demonic refinement algebra (DRA)* given at the end of [28]. That algebra is an axiomatic algebraic system for dealing with predicate transformers under a demonic view of non-determinacy.

Besides τ (which is characterised by the domain axioms of [8]) DRA has an enabledness operator ϵ , defined not in terms of tests but by dual axioms in terms of *guards* or assumptions. These take the form $\neg p \cdot \top + 1$ where $\top$ is the greatest element (which always exists in DRA). The intuitive meaning of tests and assumptions is briefly elaborated in the Appendix.

Let us see what assumptions (also called *guards*) are in $\text{ISO}(B)$:

$$(\langle \neg p \rangle \circ \top \vee id)(q) = \langle \neg p \rangle(\top(q)) \vee q = \langle \neg p \rangle 1 \vee q = \neg p \vee q = [p]q .$$

Written in point-free style, $\langle \neg p \rangle \circ \top \vee id = [p]$. So in $\text{ISO}(B)$ the assumptions are the de Morgan duals of the tests.

For the dual of the domain we obtain

$$(\ulcorner f)^d = \langle f(1) \rangle^d = [f(1)] = [f(\neg 0)] = [\neg f^d(0)] . \quad (12)$$

This latter expression coincides with that for $\epsilon(f^d)$ in the mentioned concrete model of [28], so that by $(g^d)^d = g$ we have the equation $\tau f = (\epsilon(f^d))^d$. Finally, it should be noted that the rightmost expression in (12) also corresponds to the *guard* $\neg \text{wp}(a, \text{false})$ of [24], while that for τ coincides with the termination predicate $\text{wp}(a, \text{true})$ there.

10 Conclusion and Outlook

We have shown that modal semirings and Kleene algebras form a comprehensive and flexible framework for handling various modal logics in a uniform algebraic fashion. We therefore think that the design of new modal systems geared toward special applications may benefit from using this algebraic approach.

An interesting approach, close in spirit, is [4], where modules over quantales are used to define an algebraic semantics of modal operators. However, having separate sorts for actions and (the equivalent) test makes that framework less flexible than ours, since those entities cannot be combined freely with the same operators. Moreover, the restriction to (full) quantales is less general than what the semiring framework offers.

One topic we have omitted from the present paper is that of infinite iteration. This has been treated in [21]. However, there is a restriction. Although over a complete Boolean algebra B infinite iteration can be defined as $f^\omega =_{df} \nu g . f \circ g$ in $\text{ISO}(B)$, this does not imply the usual omega coinduction law $c \leq a \cdot c + b \Rightarrow c \leq a^\omega + a^* \cdot b$ [7]. It only does so in $\text{DIS}(B)$. However, as stated in [26], disjunctivity is not a natural requirement for games.

Other future work will concern the proper treatment of infinite iteration of games, further applications (e.g., extending the work on characterisation of winning strategies in [2] and of winning and losing positions in [9]), but also partial mechanisation of the (largely equational and fully first-order) axiomatic system. First steps into the latter direction using the tools Prover9 and Mace4 [20] have been taken by P. Höfner and G. Struth at Sheffield [13].

Acknowledgments. I am grateful to E. André for drawing my attention to the area of modal agent logics and to B. Dill, R. Glück, P. Höfner, H. Leiß, M.E. Müller, K. Solin and the referees for helpful comments and suggestions.

References

1. Back, R.J., von Wright, J.: Refinement calculus — A systematic introduction. Springer, Heidelberg (1998)
2. Backhouse, R., Michaelis, D.: Fixed-point characterisation of winning strategies in impartial games. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS 2003. LNCS, vol. 3051, pp. 34–47. Springer, Heidelberg (2004)

3. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Proc. 7th conference on Theoretical Aspects of Rationality and Knowledge, Evanston, Illinois, pp. 43–56 (1998)
4. Baltag, A., Coecke, B., Sadrzadeh, M.: Epistemic actions as resources. *J. Log. Comput.* 17, 555–585 (2007)
5. van Benthem, J., Liu, F.: Dynamic logic of preference upgrade. *J. Applied Non-Classical Logics* 2006 (manuscript, 2004) (to appear)
6. Bergstra, J.A., Fokkink, W., Ponse, A.: Process algebra with recursive operations. In: Bergstra, J.A., Smolka, S., Ponse, A. (eds.) *Handbook of process algebra*, pp. 333–389. North-Holland, Amsterdam (2001)
7. Cohen, E.: Separation and reduction. In: Backhouse, R., Oliveira, J.N. (eds.) *MPC 2000. LNCS*, vol. 1837, pp. 45–59. Springer, Heidelberg (2000)
8. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. Institute of Computer Science, University of Augsburg, Technical Report 2003-7. Revised version: *ACM Transaction on Computational Logic* 7(4), 798–833 (2006)
9. Desharnais, J., Möller, B., Struth, G.: Modal Kleene algebra and applications — A survey. *Journal on Relational Methods in Computer Science* 1, 93–131 (2004)
10. Desharnais, J., Möller, B., Struth, G.: Termination in modal Kleene algebra. In: Lévy, J.-J., Mayr, E., Mitchell, J. (eds.) *Exploring new frontiers of theoretical informatics. IFIP Series*, vol. 155, pp. 653–666. Kluwer, Dordrecht (2006), Extended version: Institute of Computer Science, University of Augsburg, Technical Report 2006-23
11. Dijkstra, E.: *A discipline of programming*. Prentice-Hall, Englewood Cliffs (1976)
12. Harel, D., Kozen, D., Tiuryn, J.: *Dynamic logic*. MIT Press, Cambridge (2000)
13. Höfner, P., Struth, G.: Automated reasoning in Kleene algebra. In: Pfenning, F. (ed.) *CADE 2007. LNCS (LNAI)*, vol. 4603, pp. 279–294. Springer, Heidelberg (2007)
14. Huth, M., Ryan, M.: *Logic in computer science — Modelling and reasoning about systems*, 2nd edn. Cambridge University Press, Cambridge (2004)
15. Jónsson, B., Tarski, A.: Boolean algebras with operators, Part I. *American Journal of Mathematics* 73, 891–939 (1951)
16. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.* 110(2), 366–390 (1994)
17. Kozen, D.: Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems* 19(3), 427–443 (1997)
18. Kuich, W., Salomaa, A.: Semirings, automata, languages. In: *EATCS Monographs on Theoretical Computer Science*, vol. 5, Springer, Heidelberg (1986)
19. McCarthy, J.: Formalization of two puzzles involving knowledge, <http://www-formal.stanford.edu/jmc/puzzles/puzzles.html>
20. McCune, W.: Prover9 and Mace4, <http://www.cs.unm.edu/~mccune/mace4/>
21. Möller, B.: Lazy Kleene algebra. In: Kozen, D. (ed.) *MPC 2004. LNCS*, vol. 3125, pp. 252–273. Springer, Heidelberg (2004) Revised Version: B. Möller: Kleene getting lazy. *Science of Computer Programming* (in press)
22. Möller, B., Höfner, P., Struth, G.: Quantales and temporal logics. In: Johnson, M., Vene, V. (eds.) *AMAST 2006. LNCS*, vol. 4019, pp. 263–277. Springer, Heidelberg (2006)
23. Möller, B., Struth, G.: Algebras of modal operators and partial correctness. *Theoretical Computer Science* 351, 221–239 (2006)
24. Nelson, G.: A generalization of Dijkstra’s calculus. *ACM Transactions on Programming Languages and Systems* 11, 517–561 (1989)

25. Parikh, R.: Propositional logics of programs: new directions. In: Karpinski, M. (ed.) FCT 1983. LNCS, vol. 158, pp. 347–359. Springer, Heidelberg (1983)
26. Pauly, M., Parikh, R.: Game logic – An overview. *Studia Logica* 75, 165–182 (2003)
27. Solin, K.: Dynamic epistemic semirings. Institute of Computer Science, University of Augsburg, Technical Report, 2006 (June 17, 2006)
28. Solin, K., von Wright, J.: Refinement algebra with operators for enabledness and termination. In: Uustalu, T. (ed.) MPC 2006. LNCS, vol. 4014, pp. 397–415. Springer, Heidelberg (2006)
29. Wikipedia: Unexpected hanging paradox,
http://en.wikipedia.org/wiki/Unexpected_hanging_paradox

Appendix

First we prove an auxiliary lemma about relative complements.

Lemma A. Assume in a Boolean algebra $r \leq p \wedge q \wedge s \leq p \wedge \neg q \wedge r \vee s = p$. Then $r = p \wedge q \wedge s = p \wedge \neg q$.

Proof. Observe that $s \wedge q \leq p \wedge \neg q \wedge q = p \wedge 0 = 0$, i.e., $s \wedge q = 0$. Hence $p \wedge q = (r \vee s) \wedge q = r \wedge q \vee s \wedge q = r \wedge q \leq r$, which shows $r = p \wedge q$. Symmetrical reasoning applies to s . $\square$

Now we can give the

Proof of Lemma 9.1

1. ($\Leftarrow$) By definition, $f \leq id$. A straightforward calculation shows that the complement of f relative to id is $g(p) =_{df} p \wedge \neg f(1)$.
 ($\Rightarrow$) Let $g \in \text{ISO}(B)$ be the complement of $f \leq id$ relative to id , i.e., $f \vee g = id$ and $f \wedge g = \mathbf{0}$. First, $f \leq id$ means $f(p) \leq p$. Second, $f \in \text{ISO}(B)$ means $f(p) \leq f(1)$. Hence $f(p) \leq p \wedge f(1)$. From $f \vee g = id$ we conclude $g(1) = \neg f(1)$ and hence, by symmetrical reasoning, $g(p) \leq p \wedge \neg f(1)$. Since

$$\begin{aligned} p \wedge f(1) \vee p \wedge \neg f(1) &= p \wedge (f(1) \vee \neg f(1)) = p \wedge 1 = p, \\ p \wedge f(1) \wedge p \wedge \neg f(1) &= p \wedge f(1) \wedge \neg f(1) = p \wedge 0 = 0, \end{aligned}$$

we obtain $f(p) = p \wedge f(1)$ and $g(p) = p \wedge \neg f(1)$ by Lemma A.

2. By (4) and 1. we have for $f \in \text{test}(\text{ISO}(B))$ that $f = \langle f(1) \rangle$, which shows ($\subseteq$). The reverse inclusion is immediate from isotony of $\langle p \rangle$. $\square$

We conclude by explaining the relation between tests and assumptions. We first introduce a test-based conditional as if p then a else $b \Leftrightarrow_{df} p \cdot a + \neg p \cdot b$. With its help assertions and assumptions can be defined as

$$\text{assert } p =_{df} \text{if } p \text{ then } 1 \text{ else } 0 \qquad \text{assume } p =_{df} \text{if } p \text{ then } 1 \text{ else } \top,$$

the latter provided S has a greatest element $\top$. In an operational view, both constructs check whether p holds at the time of their execution. If so, they simply proceed (remember that 1 stands for the null action). If not, the assertion aborts while the assumption may do anything ($\top$ means the set of all possible choices, so we have the behaviour *ex falso quodlibet*).

Both expressions can be simplified. For assertions we obtain

$$\mathbf{assert} \ p = p \cdot 1 + \neg p \cdot 0 = p + 0 = 0 .$$

Hence the construct **assert** p could be omitted; we have introduced it just for symmetry. For assumptions we get, since $\neg p \cdot 1 \leq \neg p \cdot \top$,

$$\begin{aligned} \mathbf{assume} \ p &= p \cdot 1 + \neg p \cdot \top = p \cdot 1 + \neg p \cdot 1 + \neg p \cdot \top \\ &= (p + \neg p) \cdot 1 + \neg p \cdot \top = 1 + \neg p \cdot \top , \end{aligned}$$

which is the expression given in Section [9](#).

Theorem Proving Modulo Based on Boolean Equational Procedures

Camilo Rocha and José Meseguer

Department of Computer Science
University of Illinois at Urbana-Champaign
201 N Goodwin Ave
Urbana, IL 61801
{hrochan2,meseguer}@cs.uiuc.edu

Abstract. Deduction with inference rules *modulo* computation rules plays an important role in automated deduction as an effective method for scaling up. We present four equational theories that are isomorphic to the traditional Boolean theory and show that each of them gives rise to a Boolean decision procedure based on a canonical rewrite system modulo associativity and commutativity. Then, we present two modular extensions of our decision procedure for Dijkstra-Scholten propositional logic to the Sequent Calculus for First Order Logic and to the Syllogistic Logic with Complements of L. Moss. These extensions take the form of *rewrite theories* that are sound and complete for performing *deduction modulo* their equational parts and exhibit good mechanization properties. We illustrate the practical usefulness of this approach by a direct implementation of one of these theories in Maude rewriting logic language, and automatically proving a challenge benchmark in theorem proving.

1 Introduction

The key challenge in automated deduction is *scaling up*. For the large proof efforts involved in non-toy mathematical and system verification proofs it is essential to raise the level of abstraction, so that the person performing the proofs can delegate large chunks of the effort to automated proof assistants. This need is widely felt, and approaches to meet it take different guises, such as the growing support for decision procedures, the autarkic/skeptical distinction between proofs and computations [2], and the so-called “deduction modulo” approach [7], which, as shown by Viry [28], is very closely related to the use of rewriting logic as a logical framework [18], so that the distinction between computation and deduction is captured by the corresponding distinction between equations and rules in a rewrite theory $\mathcal{R}_{\mathcal{L}}$ formalizing the inference system of the given logic $\mathcal{L}$.

Specifically, the rewrite theory $\mathcal{R}_{\mathcal{L}}$ is a triple $\mathcal{R}_{\mathcal{L}} = (\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}}, R_{\mathcal{L}})$, where: (i) $\Sigma_{\mathcal{L}}$ is a signature describing the syntax of the logic $\mathcal{L}$; (ii) $E_{\mathcal{L}}$ is a set of confluent and terminating *equations* modulo $A_{\mathcal{L}}$, corresponding to those parts of the

deduction process that, being deterministic, can be safely automated as *computation rules* without any proof search; and (iii) $R_{\mathcal{L}}$ is a, typically small, set of rewrite rules capturing those essentially nondeterministic aspects of logical inference in $\mathcal{L}$ which require proof search. Both the computation rules $E_{\mathcal{L}}$ and the deduction rules $R_{\mathcal{L}}$ are executed by rewriting *modulo* a set $A_{\mathcal{L}}$ of equations specifying some *structural axioms* in $\mathcal{L}$ such as, for example, the associativity and commutativity of an addition operator $+$ at the level of *terms*, or of a conjunction operator at the level of *formulas*, or the similar associativity and commutativity of the formula union operator (typically denoted with the symbol $\sqcup$) in a set of formulas $\Gamma = A_1, \dots, A_n$ at the level of *sequents*. In a traditional inference system, *all* these tasks —now delegated to either $E_{\mathcal{L}}$, or $A_{\mathcal{L}}$, or $R_{\mathcal{L}}$ — would be performed as *deduction* tasks, which gets the deduction process bogged down in endless minutiae, and misses countless opportunities of making proof generation much more efficient by identifying and exploiting its computational subtasks. The point, of course, is that although both $E_{\mathcal{L}}$ and $R_{\mathcal{L}}$ are executed by rewriting, $E_{\mathcal{L}}$, being confluent and terminating, has a single outcome in the form of a so-called simplified or canonical form, and can be executed as it were “blindly,” without any search, and therefore also blindingly fast and with typically modest memory requirements. Furthermore, $A_{\mathcal{L}}$ provides yet one more level of computational automation in the form of $A_{\mathcal{L}}$ -matching or $A_{\mathcal{L}}$ -unification algorithms.

By “deduction modulo” in this context, what we then mean is that the inference rules $R_{\mathcal{L}}$ are really operating not at the level of syntactic entities as in the traditional case, but *modulo* the entire equational theory $(\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$, comprising both the computation rules $E_{\mathcal{L}}$ and the structural axioms $A_{\mathcal{L}}$. Therefore, one step of inference with $R_{\mathcal{L}}$ modulo $E_{\mathcal{L}} \cup A_{\mathcal{L}}$ may literally correspond to millions of inference steps in a traditional inference system for $\mathcal{L}$.

These ideas have been illustrated in detail for many logics in various papers, including, for example, various sequent calculi in [18], the “sequent calculus modulo” of G. Dowek, T. Hardin and C. Kirchner in [7], Viry’s rewrite theory for the sequent calculus of first-order logic in [28], and the representation of pure type systems in rewriting logic in [26]. In this paper we concentrate our attention on what we think is an interesting instance of the deduction modulo idea that combines two obvious strengths: (i) the general power of the deduction modulo framework; and (ii) the intrinsic power of equationally-based Boolean *decision procedures* operating at the level of formulas. The idea, therefore, is that the equational theory $(\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$ we are reasoning modulo, includes a confluent and terminating subtheory $(\Sigma_{\text{BOOL}}, E_{\text{BOOL}} \cup A_{\text{BOOL}}) \subseteq (\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$, where $(\Sigma_{\text{BOOL}}, E_{\text{BOOL}} \cup A_{\text{BOOL}})$ provides a decision procedure for Boolean equivalence of formulas in $\mathcal{L}$. This can be very useful, because other equations in $E_{\mathcal{L}}$ (operating, for example at the level of sequents) or some rules in $R_{\mathcal{L}}$, may immediately take advantage of the fact that we have simplified a formula to a tautology or a falsity to finish off a whole deduction subgoal.

Specifically, in Sections 3 and 4 we discuss in detail four such equationally-based Boolean decision procedures. One is the well-known procedure due to J. Hsiang, who gave a confluent and terminating set of equations for the theory

of Boolean rings *modulo* associativity and commutativity in his UIUC Ph.D. thesis [14]. The other three are, to the best of our knowledge, new. We characterize their soundness and completeness by the satisfaction of two key properties: (i) they are all isomorphic to the standard Boolean theory; and (ii) they are all confluent and terminating modulo some associativity and commutativity axioms.

In this paper we give particular attention to one of these four Boolean theories, namely, a decision procedure for the propositional fragment of the Dijkstra-Scholten logic [6]. This logic has been shown by Dijkstra and Scholten to be very useful in program correctness proofs in the Dijkstra style, and has attracted a substantial following in research, teaching and programming, including [6, 11, 16]. It has the same expressive power as standard first-order logic [16]; and includes an interesting propositional fragment [12]. However, to the best of our knowledge this logic has not yet been mechanized, and no *equational* decision procedure based on confluent and terminating equations was known for it. The obvious approach to obtain a scalable mechanization of the Dijkstra-Scholten (first-order) logic in a “deduction modulo” style is then to specify it as a rewrite theory $\mathcal{R}_{DS} = (\Sigma_{DS}, E_{DS} \cup A_{DS}, R_{DS})$, where $(\Sigma_{DS}, E_{DS} \cup A_{DS})$ includes the just-mentioned, equationally-based decision procedure for the Boolean equivalence of formulas. We do just that, in the form of a Dijkstra-Scholten-style sequent calculus for first-order logic that we prove sound and complete in Section 5. We also show in Section 5.1 that the rewrite theory $\mathcal{R}_{DS}$ satisfies all the essential requirements for being *executable* by rewriting by showing that: (i) the equational axioms A_{DS} consist only of associativity and commutativity axioms for which A_{DS} -matching and A_{DS} -unification algorithms are readily available; (ii) the equations E_{DS} , comprising not only the equations of our decision procedure but also logical equivalences at the level of sequents, are *confluent and terminating* modulo A_{DS} , and (iii) the inference rules R_{DS} are *weakly coherent* with respect to the equations E_{DS} modulo A_{DS} , which means that we can always execute the rules in R_{DS} *after* all goals have been simplified by E_{DS} without any loss in logical completeness. In Section 5.2 we illustrate the practical usefulness of this approach by a direct implementation of the rewrite theory $\mathcal{R}_{DS}$ in the Maude rewriting logic language that is able to prove automatically a challenge benchmark in theorem proving, namely, Andrews’ challenge [10].

As further evidence for the power of the deduction modulo approach to theorem proving supported by rewriting logic, we summarize in Section 6 another case study developed more fully in [22], namely, a Dijkstra-Scholten-style decision procedure for the Syllogistic Logic with Complements of L. Moss [20]. In this, simpler case, no proof search is involved at all, that is, all is “computation,” and there is no “deduction,” so that the set of rules is empty and the entire decision procedure for this logic takes the form of an *equational* theory extending that of the equational theory for proposition Dijkstra-Scholten logic. We conclude the paper with some final remarks and a discussion of future work. For detailed proofs, complete specifications, and further discussion on the results presented in this paper, we refer the reader to the technical report [23].

2 Rewrite Theories and Weak Coherence

The reason why rewriting logic directly captures the “theorem proving modulo” idea is that, given a rewrite theory of the form $\mathcal{R} = (\Sigma, E \cup A, R)$, where A is a set of “structural” equational axioms (typically associativity and/or commutativity and/or identity) such that there exists a *matching algorithm modulo A* producing a finite number of A -matching substitutions, or failing otherwise, then rewriting with rules R in $\mathcal{R}$ takes place *modulo* $E \cup A$. For example, if $\mathcal{R} = \mathcal{R}_{\mathcal{L}}$ is the rewrite theory of a sequent calculus for $\mathcal{L}$, a sequent is a term t , but the rules R in $\mathcal{R}$ do not rewrite just sequents: they rewrite $E \cup A$ -equivalence class $[t]_{E \cup A}$ in the free algebra on variables X modulo $E \cup A$, denoted $T_{\Sigma/E \cup A}(X)$. More precisely, we have a one-step rewrite $[t]_{E \cup A} \rightarrow_{\mathcal{R}} [t']_{E \cup A}$ in $\mathcal{R}$ iff we can find a term $u \in [t]_{E \cup A}$ such that u can be rewritten to v using some rule $l : q \rightarrow r$ in R in the standard way (see [5]), denoted $u \rightarrow_R v$, and we furthermore have $v \in [t']_{E \cup A}$. The problem is that for arbitrary E and R , whether $[t]_{E \cup A} \rightarrow_{\mathcal{R}} [t']_{E \cup A}$ holds is in general *undecidable*, even when the equations E are confluent and terminating modulo A . Therefore, the most useful rewrite theories satisfy additional executability conditions, explained below, under which we can reduce the relation $[t]_{E \cup A} \rightarrow_{\mathcal{R}} [t']_{E \cup A}$ to simpler forms of rewriting just modulo A , where both equality modulo A and matching modulo A are decidable.

The first condition is that E should be *ground confluent and terminating modulo A* [5]. This means that in the rewrite theory $\mathcal{R}_{E/A} = (\Sigma, A, E)$: (i) all rewrite sequences terminate, that is, there are no infinite sequences of the form $[t_1]_A \rightarrow_{\mathcal{R}_{E/A}} [t_2]_A \cdots [t_n]_A \rightarrow_{\mathcal{R}_{E/A}} [t_{n+1}]_A \cdots$, and (ii) for each $[t]_A \in T_{\Sigma/A}$ there is a *unique* A -equivalence class $[\text{can}_{E/A}(t)]_A \in T_{\Sigma/A}$ called the *E-canonical form* of $[t]_A$ modulo A such that there exists a terminating sequence of zero, one, or more steps $[t]_A \rightarrow_{\mathcal{R}_{E/A}}^* [\text{can}_{E/A}(t)]_A$.

The second condition is that the rules R should be *coherent* relative to the equations E modulo A [28]. This precisely means that, if we decompose the rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$ into the simpler theories $\mathcal{R}_{E/A} = (\Sigma, A, E)$ and $\mathcal{R}_{R/A} = (\Sigma, A, R)$ (which have decidable rewrite relations $\rightarrow_{\mathcal{R}_{E/A}}$ and $\rightarrow_{\mathcal{R}_{R/A}}$ because of the assumptions on A), then for each A -equivalence class $[t]_A$ such that $[t]_A \rightarrow_{\mathcal{R}_{R/A}} [t']_A$ we can always find a corresponding rewrite $[\text{can}_{E/A}(t)]_A \rightarrow_{\mathcal{R}_{R/A}} [t'']_A$ such that $[\text{can}_{E/A}(t')]_A = [\text{can}_{E/A}(t'')]_A$.

Intuitively, coherence means that we can always adopt the strategy of first simplifying a term to canonical form with E modulo A , and then apply a rule with R modulo A to achieve the effect of rewriting with R modulo $E \cup A$. The coherence condition can be relaxed to *weak coherence* of R relative to the equations E modulo A [28], where we just require that whenever $[t]_A \rightarrow_{\mathcal{R}_{R/A}} [t']_A$ we can always find a sequence of zero, one or more rewrites $[\text{can}_{E/A}(t)]_A \rightarrow_{\mathcal{R}_{R \cup E/A}}^* [t'']_A$ such that $[\text{can}_{E/A}(t')]_A = [\text{can}_{E/A}(t'')]_A$.

When formalizing a logic $\mathcal{L}$ as a rewrite theory $\mathcal{R}_{\mathcal{L}}$ one has two different options (backwards or forwards) for expressing an inference rule as a rewrite rule. We will adopt the backwards reasoning option, which rewrites the goal one wants to prove to its premise subgoals. For example, a sequent rule for

disjunction $\frac{\Gamma, B \vdash \Delta \quad \Gamma, C \vdash \Delta}{\Gamma, B \vee C \vdash \Delta}$ will be expressed as a rewrite rule $\Gamma, B \vee C \vdash \Delta \longrightarrow \Gamma, B \vdash \Delta \bullet \Gamma, C \vdash \Delta$, where $\bullet$ is an associative commutative operator denoting set union of sequents.

3 Five Isomorphic Boolean Theories

In this section we present five isomorphic equational theories, one of them the traditional Boolean theory. We structure each of these theories in the form $(\Sigma, E \cup A)$, with A some associativity and commutativity axioms.

The axiomatization of the traditional Boolean theory T_{BOOL} is that of a complemented distributive lattice.

Definition 1. *The equational theory $T_{\text{BOOL}} = (\Sigma_{\text{BOOL}}, E_{\text{BOOL}} \cup A_{\text{BOOL}})$ is given by:*

$$\begin{aligned} \Sigma_{\text{BOOL}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \neg^{(1)}, \wedge^{(2)}, \vee^{(2)}\} \\ A_{\text{BOOL}} &= \{P \wedge (Q \wedge R) = (P \wedge Q) \wedge R, P \wedge Q = Q \wedge P, \\ &\quad P \vee (Q \vee R) = (P \vee Q) \vee R, P \vee Q = Q \vee P\} \\ E_{\text{BOOL}} &= \{P \wedge P = P, P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R), \\ &\quad P \vee P = P, P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R), \\ &\quad P \wedge (P \vee Q) = P, P \vee (P \wedge Q) = P, \\ &\quad P \wedge \neg P = \mathsf{F}, P \vee \neg P = \mathsf{T}\}. \end{aligned}$$

The axioms in A_{BOOL} express the associativity and commutativity properties (AC) of the binary operators in Σ_{BOOL} . The set of equations E_{BOOL} define both $\wedge$ and $\vee$ to be idempotent, to distribute over each other and to follow the absorption laws. The last two equations in E_{BOOL} are the well-known laws of complements, the first being the definition of contradiction and the second that of the excluded middle.

We introduce the remaining four equational theories, namely, $T_{\text{DS}}, T_{\text{BR}}, T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, respectively. The theory T_{DS} is our axiomatization, as a set of confluent and terminating equations modulo AC, of the Dijkstra-Scholten propositional logic [6]. The theory T_{BR} is the theory of Boolean rings and is based on the isomorphism between Boolean algebras and Boolean rings discovered by M. H. Stone [15, 24]. As a rewrite system, T_{BR} was proposed by J. Hsiang [14] in the 1980's as a decision procedure for propositional logic. We are not aware of earlier equational presentations of $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, so we use their main function symbols as acronyms.

Definition 2. *The equational theories $T_{\text{DS}} = (\Sigma_{\text{DS}}, E_{\text{DS}} \cup A_{\text{DS}})$, $T_{\text{BR}} = (\Sigma_{\text{BR}}, A_{\text{BR}} \cup E_{\text{BR}})$, $T_{\wedge/\equiv} = (\Sigma_{\wedge/\equiv}, E_{\wedge/\equiv} \cup A_{\wedge/\equiv})$ and $T_{\vee/\oplus} = (\Sigma_{\vee/\oplus}, E_{\vee/\oplus} \cup A_{\vee/\oplus})$ are defined as follows:*

$$\begin{aligned} \Sigma_{\text{DS}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \vee^{(2)}, \equiv^{(2)}\} \\ A_{\text{DS}} &= \{P \equiv (Q \equiv R) = (P \equiv Q) \equiv R, P \equiv Q = Q \equiv P \\ &\quad P \vee (Q \vee R) = (P \vee Q) \vee R, P \vee Q = Q \vee P\} \\ E_{\text{DS}} &= \{P \equiv \mathsf{T} = P, P \equiv P = \mathsf{T}, P \vee \mathsf{T} = \mathsf{T}, P \vee \mathsf{F} = P, P \vee P = P \\ &\quad P \vee (Q \equiv R) = (P \vee Q) \equiv (P \vee R)\}, \end{aligned}$$

$$\begin{aligned}
\Sigma_{\text{BR}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \wedge^{(2)}, \oplus^{(2)}\} \\
A_{\text{BR}} &= \{P \oplus (Q \oplus R) = (P \oplus Q) \oplus R, P \oplus Q = Q \oplus P \\
&\quad P \wedge (Q \wedge R) = (P \wedge Q) \wedge R, P \wedge Q = Q \wedge P\} \\
E_{\text{BR}} &= \{P \oplus \mathsf{F} = P, P \oplus P = \mathsf{F}, P \wedge \mathsf{F} = \mathsf{F}, P \wedge \mathsf{T} = P, P \wedge P = P \\
&\quad P \wedge (Q \oplus R) = (P \wedge Q) \oplus (P \wedge R)\}, \\
\\
\Sigma_{\wedge/\equiv} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \wedge^{(2)}, \equiv^{(2)}\} \\
A_{\wedge/\equiv} &= \{P \equiv (Q \equiv R) = (P \equiv Q) \equiv R, P \equiv Q = Q \equiv P \\
&\quad P \wedge (Q \wedge R) = (P \wedge Q) \wedge R, P \wedge Q = Q \wedge P\} \\
E_{\wedge/\equiv} &= \{P \equiv \mathsf{T} = P, P \equiv P = \mathsf{T}, P \wedge \mathsf{T} = P, P \wedge \mathsf{F} = \mathsf{F}, P \wedge P = P \\
&\quad P \wedge (Q \equiv R) = (P \wedge Q) \equiv (P \wedge R) \equiv P\}, \\
\\
\Sigma_{\vee/\oplus} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \vee^{(2)}, \oplus^{(2)}\} \\
A_{\vee/\oplus} &= \{P \oplus (Q \oplus R) = (P \oplus Q) \oplus R, P \oplus Q = Q \oplus P \\
&\quad P \vee (Q \vee R) = (P \vee Q) \vee R, P \vee Q = Q \vee P\} \\
E_{\vee/\oplus} &= \{P \oplus \mathsf{F} = P, P \oplus P = \mathsf{F}, P \vee \mathsf{T} = \mathsf{T}, P \vee \mathsf{F} = \mathsf{F}, P \vee P = P \\
&\quad P \vee (Q \oplus R) = (P \vee Q) \oplus (P \vee R) \oplus P\}.
\end{aligned}$$

The function symbols $\equiv$ and $\oplus$ denote equivalence and discrepancy, respectively, and have less binding power than any other function symbol. Both symbols are associative and commutative in the theories where they are defined. The other function symbols correspond to those of Σ_{BOOL} ; we have chosen not to change their notation in order to keep the definitions and proofs as compact as possible. The symbol $\oplus$ is sometimes denoted by $\neq$ and it is known as either the symmetric difference operator in algebra or as the exclusive or operator in switching theory.

To show that the theories T_{BOOL} , T_{DS} , T_{BR} , $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$ are all isomorphic, we make precise the notion of equational *theory isomorphism*, and more generally, that of *theory morphism* in the category **Th** of equational theories (see [21]). Here we just summarize the basic idea by pointing out that a theory morphism $H : (\Sigma, E) \longrightarrow (\Sigma', E')$ maps each $f \in \Sigma_n$ to a Σ' -term with n variables and satisfies the property that if $u = v \in E$, then $E' \vdash H(u) = H(v)$.

Definition 3. *The nine morphisms appearing in Fig. 1 are defined as follows:*

- G maps identically T , F and $\vee$. For $\neg$ and $\wedge$ we have: $G(\neg P) = P \equiv \mathsf{F}$ and $G(P \wedge Q) = P \equiv Q \equiv P \vee Q$.
- G^{-1} maps identically T , F and $\vee$. For $\equiv$ we have $G^{-1}(P \equiv Q) = (P \vee \neg Q) \wedge (\neg P \vee Q)$.
- H maps identically T , F and $\wedge$. For $\neg$ and $\vee$ we have: $H(\neg P) = P \oplus \mathsf{T}$ and $H(P \vee Q) = P \oplus Q \oplus P \wedge Q$.
- H^{-1} maps identically T , F and $\wedge$. For $\oplus$ we have $H^{-1}(P \oplus Q) = (P \vee Q) \wedge (\neg P \vee \neg Q)$.
- K maps identically T , F and $\wedge$. For $\neg$ and $\vee$ we have: $K(\neg P) = P \equiv \mathsf{F}$ and $K(P \vee Q) = P \equiv Q \equiv P \wedge Q$.
- K^{-1} maps identically T , F and $\wedge$. For $\equiv$ we have $K^{-1}(P \equiv Q) = (P \wedge Q) \vee (\neg P \wedge \neg Q)$.

- L maps identically $\top$, F and $\vee$. For $\neg$ and $\wedge$ we have: $L(\neg P) = P \oplus \top$ and $L(P \wedge Q) = P \oplus Q \oplus P \vee Q$.
- L^{-1} maps identically $\top$, F and $\vee$. For $\oplus$ we have $L^{-1}(P \oplus Q) = (P \wedge \neg Q) \vee (\neg P \wedge Q)$.
- op is the duality morphism for Boolean algebras, mapping $\top$ to F , F to $\top$, $\neg$ to $\neg$, $\wedge$ to $\vee$ and $\vee$ to $\wedge$.

Theorem 1. *The morphisms op , G , H , K and L are theory isomorphisms between the corresponding theories.*

We call these isomorphisms *Boolean isomorphisms*. They give rise to new Boolean isomorphisms by composition among them. Figure 2 highlights two particular ones, namely $G \circ op \circ H^{-1}$ and $L \circ op \circ K^{-1}$, which show that the theories T_{BR} and T_{DS} , and the theories $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$ are pairs of *dual theories*. These morphisms are used in the next section to build decision procedures for propositional logic by rewriting using the theories T_{DS} , T_{BR} , $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$.

4 Four Equational Decision Procedures

In this section we explain in more detail a decision procedure for propositional logic for the equational theory T_{DS} . The exact same construction applies to T_{BR} (where it is well-known since [14]), to $T_{\wedge/\equiv}$ and to $T_{\vee/\oplus}$. The complete set of four decision procedures for propositional logic we have studied using this approach, each containing equations for all other Boolean connectives as definitional extensions, can be found in [21].

Theorem 2. *The equations E_{DS} in T_{DS} are confluent and terminating modulo A_{DS} . Similarly, the equations in $E_{\wedge/\equiv}$ and $E_{\vee/\oplus}$, in $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, are confluent and terminating modulo $A_{\wedge/\equiv}$ and $A_{\vee/\oplus}$, respectively.*

We focus on T_{DS} and refer to [21] for $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$. Termination and confluence modulo A_{DS} can be established mechanically by using formal tools that: (i) find a well-founded ordering $\succ$ on A_{DS} -equivalence classes of terms such that

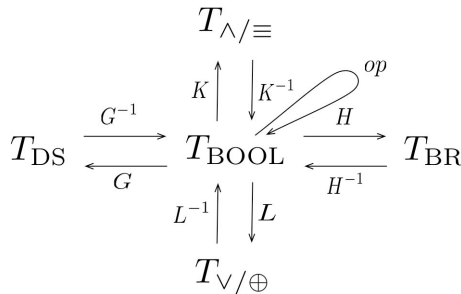


Fig. 1. Isomorphisms between the Boolean theory and the other four theories

$$\begin{array}{ccccc}
T_{\text{BR}} & \xrightarrow[\sim]{H^{-1}} & T_{\text{BOOL}} & \xleftarrow[\sim]{K^{-1}} & T_{\wedge/\equiv} \\
\downarrow \wr & & \downarrow \wr^{op} & & \downarrow \wr \\
T_{\text{DS}} & \xleftarrow[\sim]{G} & T_{\text{BOOL}} & \xrightarrow[\sim]{L} & T_{\vee/\oplus}
\end{array}$$

Fig. 2. Commutation and composition of Boolean isomorphisms

$[t]_{A_{\text{DS}}} \rightarrow_{E_{\text{DS}}/A_{\text{DS}}} [t']_{A_{\text{DS}}}$ implies $[t]_{A_{\text{DS}}} \succ [t']_{A_{\text{DS}}}$, and (ii) check confluence of E_{DS} modulo A_{DS} by computing all so-called “critical-pairs” modulo A_{DS} and showing they are all confluent. We have used the CiME tool [4] to check termination and confluence of E_{DS} modulo A_{DS} . Furthermore, it can be shown using Maude’s Sufficient Completeness Checker [13] that the canonical form of any term is either $\top$, F or $t_0 \equiv \dots \equiv t_n$, where all t_i are distinct disjunctions (modulo AC) of propositional variables (see [21] for the proof). As a consequence, we can use T_{DS} as a *decision procedure for propositional logic*. That is, we have the following equivalences for any propositional expressions t and t' :

$$T_{\text{DS}} \vdash t = t' \Leftrightarrow T_{\text{DS}} \vdash t \equiv t' = \top \Leftrightarrow \text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t] = \text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t'] .$$

In particular, since $\top$ and F are both in $E_{\text{DS}}/A_{\text{DS}}$ -canonical form, we have:

$$T_{\text{DS}} \vdash t \equiv t' = \top \Leftrightarrow \text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t \equiv t'] = [\top] \quad \text{and,}$$

$$T_{\text{DS}} \vdash t \equiv t' = \text{F} \Leftrightarrow \text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t \equiv t'] = [\text{F}].$$

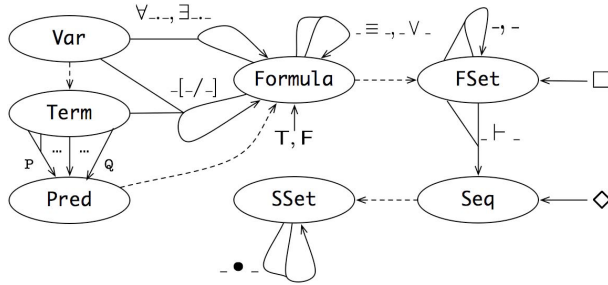
We call a proposition t a *tautology* iff $\text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t] = [\top]$ and a *falsity* iff $\text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t] = [\text{F}]$. We call t *satisfiable* iff $\text{can}_{E_{\text{DS}}/A_{\text{DS}}}[t] \neq [\text{F}]$. Therefore, our decision procedure gives also a decision procedure for checking satisfiability of any proposition t .

5 A Rewriting Modulo View of the Sequent Calculus

We present a rewrite theory $\mathcal{R}_{\text{SEQ}}^{\text{DS}} = (\Sigma_{\text{SEQ}}^{\text{DS}}, E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}, R_{\text{SEQ}}^{\text{DS}})$ modular with respect to the equational theory T_{DS} , directly inspired by the definition of the sequent calculus in [25,9]. A rewrite theory $\mathcal{R}_{\text{SEQ}}^{\text{BOOL}}$ for the sequent calculus based on the traditional connectives $\vee$, $\wedge$ and $\neg$ has been previously presented by P. Viry [27,28]. Although Viry’s equations for the formula part are executable, they fall short of being a decision procedure for Boolean equivalence of formulas. Therefore his $\mathcal{R}_{\text{SEQ}}^{\text{BOOL}}$ seems to have somewhat limited power in his “modulo” part. By contrast, our approach, by including E_{DS} in $E_{\text{SEQ}}^{\text{DS}}$, and A_{DS} in $A_{\text{SEQ}}^{\text{DS}}$, besides being readily implementable as we explain in Sections 5.1 and 5.2, has substantially more inference power in its modulo part, since any first-order formula that is a tautology or a falsity based on its Boolean structure will be automatically reduced to $\top$ or F by the E_{DS} equations, and this can be then used by the remaining equations in $E_{\text{SEQ}}^{\text{DS}}$ to automatically prove some sequents. We furthermore show in Section 5.2 the practical usefulness of our approach

by reporting on experiments with an implementation of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ in the Maude rewriting logic language.

We focus on $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ because, although first-order logic reasoning based on the Dijkstra-Scholten axiomatization has been extensively used in teaching, programming and research (see for instance [6,11,14,16]), to the best of our knowledge no mechanization of Dijkstra-Scholten-style first order logic reasoning has been developed so far, so that the implementation of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ is the first such mechanization we are aware of. However, for users interested in reasoning based on the connectives of T_{BR} , $T_{\wedge/\equiv}$ or $T_{\vee/\oplus}$, the same approach we present here for $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ can be developed in rewrite theories $\mathcal{R}_{\text{SEQ}}^{\text{BR}}$, $\mathcal{R}_{\text{SEQ}}^{\wedge/\equiv}$ and $\mathcal{R}_{\text{SEQ}}^{\vee/\oplus}$, and with the same rewriting modulo advantages. The order-sorted signature $\Sigma_{\text{SEQ}}^{\text{DS}}$ that we will use for representing terms of the sequent calculus is:



The sort **Formula** corresponds to first-order formulas built from the constants **T** and **F**, the binary operators $\equiv$ and $\vee$, and universal and existential quantification. The atomic building blocks for formulas are predicates of sort **Pred** ranging over first order terms **Term**, and constructed by predicate symbols P, Q , etc. of different arities. The sort **Var** corresponds to names of bound variables. The operator $[-/-]$ stands for explicit substitution of a variable by a term in a formula. The sort **FSet** corresponds to sets of formulas, with the constant $\square$ denoting the empty set of formulas. The sorts **Seq** and **SSet** represent first-order sequents and sets of first-order sequents, respectively. We denote the trivial sequent with the constant symbol $\diamond$. Dashed lines represent sort inclusions.

In the rest of this section we use the variables $B, C, \dots$, to represent formulas, $\Gamma, \Delta, \dots$, to represent sets of formulas, $S, S', \dots$ to represent sequents, and $SS, SS', \dots$, to represent sets of sequents.

Definition 4. The rewrite theory $\mathcal{R}_{\text{SEQ}}^{\text{DS}} = (\Sigma_{\text{SEQ}}^{\text{DS}}, E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}, R_{\text{SEQ}}^{\text{DS}})$ is defined as follows:

$$\begin{aligned}
 A_{\text{SEQ}}^{\text{DS}} &= A_{\text{FORM}} \cup \{ \Gamma, (\Delta, \Pi) = (\Gamma, \Delta), \Pi, \Gamma, \Delta = \Delta, \Gamma, \Gamma, \square = \Gamma, \\
 &\quad SS \bullet (SS' \bullet SS'') = (SS \bullet SS') \bullet SS'', SS \bullet SS' = SS' \bullet SS, \\
 &\quad SS \bullet \diamond = SS \} \\
 E_{\text{SEQ}}^{\text{DS}} &= E_{\text{SUBS}} \cup E_{\text{FORM}} \cup \{ \forall x. \mathbf{T} = \mathbf{T}, \forall x. \mathbf{F} = \mathbf{F}, \exists x. B = \forall x. (B \equiv \mathbf{F}) \equiv \mathbf{F}, \\
 &\quad \Gamma, \mathbf{F} \vdash \Delta = \diamond, \Gamma \vdash \mathbf{T}, \Delta = \diamond, \Gamma, \mathbf{T} \vdash \Delta = \Gamma \vdash \Delta, \\
 &\quad \Gamma \vdash \mathbf{F}, \Delta = \Gamma \vdash \Delta, \Gamma, \Gamma = \Gamma, SS \bullet SS = SS \}
 \end{aligned}$$

$$\begin{aligned}
R_{\text{SEQ}}^{\text{DS}} = \{ & \Gamma, B \vdash B, \Delta \longrightarrow \diamond, \\
& \Gamma, B \equiv C \vdash \Delta \longrightarrow \Gamma, B, C \vdash \Delta \bullet \Gamma \vdash B, C, \Delta, \\
& \Gamma \vdash B \equiv C, \Delta \longrightarrow \Gamma, B \vdash C, \Delta \bullet \Gamma, C \vdash B, \Delta, \\
& \Gamma, B \vee C \vdash, \Delta \longrightarrow \Gamma, B \vdash \Delta \bullet \Gamma, C \vdash \Delta, \\
& \Gamma \vdash B \vee C, \Delta \longrightarrow \Gamma \vdash B, C, \Delta, \Gamma, \forall x. B \vdash \Delta \longrightarrow \Gamma, B[t/x] \vdash \Delta \\
& \Gamma \vdash \forall x. B, \Delta \longrightarrow \Gamma \vdash B[y/x], \Delta \},
\end{aligned}$$

where A_{FORM} and E_{FORM} correspond to the equations A_{DS} and E_{DS} defined over the sort **Formula**, E_{SUBS} to the equations for explicit substitution, t is any first order term free for x and y is a variable not occurring free in Γ, B, Δ .

Equations in $A_{\text{SEQ}}^{\text{DS}}$ specify associativity, commutativity and the existence of an identity element for sets of formulas and sequents, in addition to those equations extended from A_{DS} . New equations in $E_{\text{SEQ}}^{\text{DS}}$ express different well-known logical equivalences between both formulas and sequents. The rewrite rules in $R_{\text{SEQ}}^{\text{DS}}$ correspond to a deductively complete subset of the sequent calculus rules presented in [25]. A proof of a sequent S modulo $E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}$ is then represented as a rewriting logic proof in $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ of the form $[S]_{E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}} \longrightarrow_{R_{\text{SEQ}}^{\text{DS}}}^* [\diamond]_{E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}}$, which we abbreviate as $\mathcal{R}_{\text{SEQ}}^{\text{DS}} \vdash S \longrightarrow^* \diamond$.

Theorem 3. *The rewrite theory $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ is sound and complete, that is, a sequent S is provable in the sequent calculus iff there is a derivation $\mathcal{R}_{\text{SEQ}}^{\text{DS}} \vdash S \longrightarrow^* \diamond$.*

5.1 $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ Is Weakly Coherent

As mentioned in Section 2 for a rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$ to be efficiently executable it is very important to show that its equational theory E is confluent and terminating modulo A , and its rewrite rules R are weakly coherent [28] relative to its equations E modulo the given equational axioms A . We can then *execute* both the rules R and the equations E by rewriting modulo A without losing completeness. Therefore, for our theory $\mathcal{R}_{\text{SEQ}}^{\text{DS}} = (\Sigma_{\text{SEQ}}^{\text{DS}}, E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}, R_{\text{SEQ}}^{\text{DS}})$, proofs of confluence and termination of $E_{\text{SEQ}}^{\text{DS}}$ modulo $A_{\text{SEQ}}^{\text{DS}}$, and coherence of $R_{\text{SEQ}}^{\text{DS}}$ with respect to $E_{\text{SEQ}}^{\text{DS}}$ modulo $A_{\text{SEQ}}^{\text{DS}}$, mean that $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ provides a *mechanization* of the sequent calculus modulo $E_{\text{SEQ}}^{\text{DS}} \cup A_{\text{SEQ}}^{\text{DS}}$ by rewriting. Section 5.2 discusses our experience with the mechanization of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$. Here we focus on the proofs of confluence, termination and weak coherence.

Theorem 4. *$E_{\text{SEQ}}^{\text{DS}}$ is confluent and terminating modulo $A_{\text{SEQ}}^{\text{DS}}$, and $R_{\text{SEQ}}^{\text{DS}}$ is weakly coherent with respect to $E_{\text{SEQ}}^{\text{DS}}$ modulo $A_{\text{SEQ}}^{\text{DS}}$.*

Termination and confluence of $E_{\text{SEQ}}^{\text{DS}}$ have been mechanically checked with the CiME system, assuming that the explicit substitution calculus we use is totally defined over formulas and does not generate any overlapping with the remaining equations and rules. We have checked weak coherence by checking that all critical pairs between $R_{\text{SEQ}}^{\text{DS}}$ and $E_{\text{SEQ}}^{\text{DS}}$ are properly joinable.

5.2 An Executable Specification in Maude

We present part of the specification of the rewrite theory $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ in Maude. Maude is a high-performance logical framework based on rewriting logic [3]. We only give the fragment corresponding to the sequent rewrite rules in $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$. The key point is that, since Maude modules *are* rewrite theories, the Maude specification of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ is just a transcript in typewriter font notation of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$, plus a few auxiliary functions to handle variables and substitutions.

```

mod SEQ is
...
  vars FSB FSC : FSet .    vars B C : Formula    var S : Seq .
  rl FSB,B |- B,FSC => mts .
  rl FSB,B equ C |- FSC => FSB,B,C |- FSC * FSB |- B,C,FSC .
  rl FSB |- B equ C,FSC => FSB,B |- C,FSC * FSB,C |- B,FSC .
  rl FSB,B or C |- FSC => FSB,B |- FSC * FSB,C |- FSC .
  rl FSB |- B or C,FSC => FSB |- B,C,FSC .
  rl FSB,[x : B] |- FSC => FSB,B[t/x] |- FSC [nonexec] .
  rl FSB |- [x : B],FSC => FSB |- B[newVar(FSB,B,FSC)/x],FSB .
endm

```

Universal quantification is represented with square brackets. We use `mts` to represent $\diamond$, `_equ_` for $\equiv$, `_or_` for $\vee$, and `_*_` for $\bullet$. Both `_` and `_*_` are declared as ACU operators, that is, as associative and commutative, and having an identity element. Maude efficiently implements matching and unification modulo AC and ACU. The last two rules deserve special attention. The next-to-last rule is declared non-executable (`nonexec`) because there is an extra-variable in its right-hand side, and thus the derivation tree may have infinite branching. The key observation is that the presence of extra variables in a rule's right-hand side, while making rewriting with it problematic, is unproblematic for *narrowing* with the rules of a coherent or weak coherent rewrite theory $\mathcal{R}$ modulo its equational axioms, under the assumption that its rewrite rules are *topmost*. This makes narrowing with the rules of the rewrite theory a sound and complete deduction process [19] for solving existential queries of the form $\exists \vec{x}. t(\vec{x}) \longrightarrow^* t'(\vec{x})$. In our case, the existential queries in question are of the form $\Box \vdash B \longrightarrow^* \diamond$, where B is the FOL sentence we want to prove. Although B is a sentence and therefore has no free variables, the above next-to-last rule introduces new variables, which are then incrementally instantiated as new rules are used to narrow the current set of sequents at each step. We can perform such narrowing by exploiting the efficient AC and ACU unification algorithms available in the current version of Maude and the fact that it is a reflective language [3]. The last rule makes explicit the need for the auxiliary function `newVar` to generate fresh variables not occurring in the given formulas.

We have used the complete specification in Maude of $\mathcal{R}_{\text{SEQ}}^{\text{DS}}$ to mechanically prove several FOL theorems. Here, we present the case study of Andrew's

challenge [10], a theorem that is quite difficult to prove for some theorem provers and is used as a benchmark. Andrew's challenge is to prove the following theorem:

$$(\exists x.\forall y.(P(x) \equiv P(y)) \equiv ((\exists z.Q(z)) \equiv (\forall w.P(w)))) \equiv \\ (\exists x.\forall y.(Q(x) \equiv Q(y)) \equiv ((\exists z.P(z)) \equiv (\forall w.Q(w)))) .$$

Since $\equiv$ is both associative and commutative, we can rephrase Andrew's challenge as $B \equiv C$, where:

$$\begin{aligned} B &: \exists x.\forall y.(P(x) \equiv P(y)) \equiv \exists z.P(z) \equiv \forall w.P(w) \\ C &: \exists x.\forall y.(Q(x) \equiv Q(y)) \equiv \exists z.Q(z) \equiv \forall w.Q(w) , \end{aligned}$$

and it is assumed that the formula is closed. Observe that B is an instance of C , and vice versa. Hence, it is enough to prove B or C . Here, we choose to prove the former, whose translation corresponds to the $\Sigma_{\text{SEQ}}^{\text{DS}}$ -term $\widehat{B}$:

$$\{ \text{v}(0) : [\text{v}(1) : P(\text{v}(1)) \text{ equ } P(\text{v}(2))] \} \text{ equ } \{ \text{v}(3) : P(\text{v}(3)) \} \text{ equ } [\text{v}(4) : P(\text{v}(4))]$$

where P is of sort Pred . The proof search in Maude using narrowing modulo the $A_{\text{SEQ}}^{\text{DS}}$ axioms is shown below:

```
Maude> red narrowSearch( mtf |-  $\widehat{B}$  , mts , full ACU-unify E-simplify ) .
rewrites: 49342982 in 79550ms cpu (822902ms real) (620276 rewrites/second)
```

We have used the auxiliary function `narrowSearch` which calls the narrowing strategy we use. The first argument corresponds to the sequent we want to prove, the second to the empty sequent (i.e., to the term where there is nothing left to prove) and the third to a list of parameters for the narrowing algorithm; in this case we use ACU unification and simplification with the equations before and after any narrowing step. Upon termination, the narrowing strategy returns the substitution found, meaning that the initial sequent can be transformed into the empty one and the time taken for the search.

6 Theorem Proving Modulo in Syllogistic Logic

Our tour of theorem proving modulo is not over yet. In this section we briefly summarize the results of [22] where we present the equational theory $T_{\text{CSYLL}}^{\text{DS}} = (\Sigma_{\text{CSYLL}}^{\text{DS}}, E_{\text{CSYLL}}^{\text{DS}} \cup A_{\text{CSYLL}}^{\text{DS}})$, an extension of T_{DS} , providing a decision procedure for the Syllogistic Logic with Complements of L. Moss [20]. The main feature of this sound and complete (strict) subset of Monadic First Order Logic, is the extension of the classical Syllogistic Logic with a complement operator.

We use the set Π of monadic predicates (predicates for short) $P, Q, \dots$, which in turn represent plural common nouns, to parameterize the language of Syllogistic Logic with Complements.

Definition 5. We define $\mathcal{L}(\Pi)$, for any $\pi \in \Pi$ and Atoms P and Q , as follows:

Atom $::= \pi \mid \pi^C$

Sentence $::= \text{All } P \text{ are } Q \mid \text{Some } P \text{ are } Q \mid \neg(\text{Sentence}) \mid (\text{Sentence}) \circ (\text{Sentence})$

where $\circ$ stands for any binary operator in Σ_{DS} . The semantics of the sentences and atoms is the traditional one inherited from FOL [20].

Definition 6 ([20]). Let P , Q and R be $\mathcal{L}(\Pi)$ -atoms. The inference system $K_{\mathcal{L}}$ of Syllogistic Logic with Complements is a Hilbert-style one, having modus ponens as the only inference rule, and with the following axioms:

1. All substitution instances of propositional tautologies
2. All P are P
3. $(\text{All } P \text{ are } R) \wedge (\text{All } R \text{ are } Q) \Rightarrow (\text{All } P \text{ are } Q)$
4. $(\text{All } Q \text{ are } R) \wedge (\text{Some } P \text{ are } Q) \Rightarrow (\text{Some } R \text{ are } P)$
5. $(\text{Some } P \text{ are } Q) \Rightarrow (\text{Some } P \text{ are } P)$
6. $\neg(\text{Some } P \text{ are } P) \Rightarrow (\text{All } P \text{ are } Q)$
7. $(\text{Some } P \text{ are } Q^C) \equiv \neg(\text{All } P \text{ are } Q)$.

In turn, the theory $T_{\text{CSYLL}}^{\text{DS}}$ is a many-sorted equational theory with sorts **Term** and **Sentence**.

Definition 7. The theory $T_{\text{CSYLL}}^{\text{DS}} = (\Sigma_{\text{CSYLL}}^{\text{DS}}, E_{\text{CSYLL}}^{\text{DS}} \cup A_{\text{CSYLL}}^{\text{DS}})$ is defined as follows. Its signature $\Sigma_{\text{CSYLL}}^{\text{DS}}$ has the following declarations:

T, F	: $\rightarrow$ Term
$\neg$	: Term $\rightarrow$ Term
$\equiv, \neq, \vee, \wedge, \Rightarrow, \Leftarrow$	: Term Term $\rightarrow$ Term
T, F	: $\rightarrow$ Sentence
$\neg$	: Sentence $\rightarrow$ Sentence
$\equiv, \neq, \vee, \wedge, \Rightarrow, \Leftarrow$	: Sentence Sentence $\rightarrow$ Sentence
$[-], \{-\}$	: Term $\rightarrow$ Sentence .

The axioms $A_{\text{CSYLL}}^{\text{DS}}$ correspond to the axioms in A_{DS} duplicated for both sorts **Term** and **Sentence**. That is, if we use $A_{\text{DS}}^{\text{Term}}$ and $A_{\text{DS}}^{\text{Sentence}}$ to denote the axioms A_{DS} over the sorts **Term** and **Sentence**, respectively, we have:

$$A_{\text{CSYLL}}^{\text{DS}} = A_{\text{DS}}^{\text{Term}} \cup A_{\text{DS}}^{\text{Sentence}} .$$

Similarly, if we denote with $E_{\text{DS}}^{\text{Term}}$ and $E_{\text{DS}}^{\text{Sentence}}$ the two extensions of E_{DS} over the sort **Term** and **Sentence**, respectively, we have for $P, Q : \text{Term}$:

$$E_{\text{CSYLL}}^{\text{DS}} = E_{\text{DS}}^{\text{Term}} \cup E_{\text{DS}}^{\text{Sentence}} \cup \{ [P] = \neg\{\neg P\}, \\ \{T\} = T, \{F\} = F, \{P\} \vee \{Q\} = \{P \vee Q\} \} .$$

Square brackets are used to denote universal quantification, while curly ones denote existential quantification. Observe, first, that $T_{\text{CSYLL}}^{\text{DS}}$ extends T_{DS} for its two sorts, exploiting at two different levels the power of reduction modulo. Secondly, despite the fact that Syllogistic Logic is a subset of FOL [17], neither inference rules nor explicit substitution are part of the specification: equational logic's inference system is powerful enough to handle any ‘‘syllogistic’’ deduction.

Theorem 5. $T_{\text{CSYLL}}^{\text{DS}}$ is sound and complete with respect to $\mathcal{L}(\Pi)$, that is, for a $\mathcal{L}(\Pi)$ -sentence S , $K_{\mathcal{L}} \vdash S \Leftrightarrow T_{\text{CSYLL}}^{\text{DS}} \vdash \hat{S} = \top$, where $\hat{S}$ denotes the translation of S in $\Sigma_{\text{CSYLL}}^{\text{DS}}$.

We have also shown that the set of equations $E_{\text{CSYLL}}^{\text{DS}}$ is confluent and terminating modulo $A_{\text{CSYLL}}^{\text{DS}}$. Hence, $T_{\text{CSYLL}}^{\text{DS}}$ provides a decision procedure.

7 Concluding Remarks

We have explained the general idea of how logics can be specified as rewrite theories to obtain “theorem proving modulo” proof systems that can substantially raise the level of abstraction at which an user interacts with a theorem prover and make deduction considerably more scalable. We have then focused on building in decision procedures for Boolean equivalence of formulas, and have shown how they can be seamlessly integrated within the theorem proving modulo paradigm. Specifically, we have presented three new such equationally-based procedures, and have used one of them, deciding the Dijkstra-Scholten propositional logic, to obtain an executable rewrite theory for a sequent calculus version of Dijkstra-Scholten first-order logic that can be directly used to prove nontrivial theorems. A similar “theorem proving modulo” approach to obtain a decision procedure for the Syllogistic Logic with Complements has also been summarized.

We view this work as a step forward in bringing the theorem proving modulo ideas closer to practice. However, more research is needed in terms of developing other compelling case studies for other logics and proof systems, and in terms of developing a body of generic techniques that should make it straightforward to obtain an efficient mechanization of a logic directly from a rewriting logic specification of its inference system. Such techniques should include, for example, more efficient implementations of narrowing modulo axioms, and generic libraries of tactics expressed as generic rewriting strategies in the sense of [8].

References

1. Backhouse, R.: Program Construction: Calculating Implementations from Specifications. Willey, Chichester, UK (2003)
2. Barendregt, H.P., Barendsen, E.: Autarkic computations and formal proofs. Journal of Automated Reasoning 28(3), 321–336 (2002)
3. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C. (eds.): All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007)
4. de Recherche en, L.: Informatique. The CiME System (2007), <http://cime.lri.fr/>
5. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science. Formal Methods and Semantics, ch. 6, vol. B, pp. 243–320. North-Holland, Amsterdam (1990)
6. Dijkstra, E.W., Scholten, C.S.: Predicate Calculus and Program Semantics. Springer, Heidelberg (1990)

7. Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. *J. Autom. Reasoning* 31(1), 33–72 (2003)
8. Eker, S., Martí-Oliet, N., Meseguer, J., Verdejo, A.: Deduction, strategies, and rewriting. In: Martí-Oliet, N. (ed.) *Proc. Strategies 2006, ENTCS*, pp. 417–441. Elsevier, Amsterdam (2007)
9. Girard, J.-Y.: *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science, vol. 7. Cambridge University Press, Cambridge (1989)
10. Gries, D.: A calculational proof of Andrews’s challenge. Technical Report TR96-1602, Cornell University, Computer Science (August 28, 1996)
11. Gries, D., Schneider, F.B.: A Logical Approach to Discrete Math. In: *Texts and Monographs in Computer Science*, Springer, Heidelberg (1993)
12. Gries, D., Schneider, F.B.: Equational propositional logic. *Inf. Process. Lett.* 53(3), 145–152 (1995)
13. Hendrix, J., Ohsaki, H., Meseguer, J.: Sufficient completeness checking with propositional tree automata. Technical Report UIUCDCS-R-2005-2635, University of Illinois Urbana-Champaign (2005)
14. Hsiang, J.: *Topics in automated theorem proving and program generation*. PhD thesis, University of Illinois at Urbana-Champaign (1982)
15. Jacobson, N.: *Basic algebra*, vol. I. W. H. Freeman and Co., San Francisco, Calif (1974)
16. Lifschitz, V.: On calculational proofs. *Ann. Pure Appl. Logic* 113(1-3), 207–224 (2001)
17. Łukasiewicz, J.: *Aristotle’s Syllogistic, From the Standpoint of Modern Formal Logic*. Oxford University Press, Oxford (1951)
18. Martí-Oliet, N., Meseguer, J.: Rewriting logic as a logical and semantic framework. In: Gabbay, D., Guenther, F. (eds.) *Handbook of Philosophical Logic*, 2nd. edn., pp. 1–87. Kluwer Academic Publishers, 2002. First published as SRI Tech. Report SRI-CSL-93-05 (August 1993)
19. Meseguer, J., Thati, P.: Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. *Higher-Order and Symbolic Computation* 20(1–2), 123–160 (2007)
20. Moss, L.S.: *Syllogistic logic with complements* (Draft 2007)
21. Rocha, C., Meseguer, J.: Five isomorphic Boolean theories and four equational decision procedures. Technical Report 2007-2818, University of Illinois at Urbana-Champaign (2007)
22. Rocha, C., Meseguer, J.: A rewriting decision procedure for Dijkstra-Scholten’s syllogistic logic with complements. *Revista Colombiana de Computación* 8(2) (2007)
23. Rocha, C., Meseguer, J.: Theorem proving modulo based on boolean equational procedures. Technical Report 2007-2922, University of Illinois at Urbana-Champaign (2007)
24. Simmons, G.F.: *Introduction to topology and modern analysis*. McGraw-Hill Book Co., Inc, New York (1963)
25. Socher-Ambrosius, R., Johann, P.: *Deduction Systems*. Springer, Berlin (1997)
26. Stehr, M.-O., Meseguer, J.: Pure type systems in rewriting logic: Specifying typed higher-order languages in a first-order logical framework. In: Owe, O., Kroghdahl, S., Lyche, T. (eds.) *From Object-Oriented to Formal Methods*. LNCS, vol. 2635, pp. 334–375. Springer, Heidelberg (2004)
27. Viry, P.: Adventures in sequent calculus modulo equations. *Electr. Notes Theor. Comput. Sci.* 15 (1998)
28. Viry, P.: Equational rules for rewriting logic. *Theoretical Computer Science* 285, 487–517 (2002)

Rectangles, Fringes, and Inverses

Gunther Schmidt

Institute for Software Technology, Department of Computing Science
Universität der Bundeswehr München, 85577 Neubiberg, Germany
`Gunther.Schmidt@unibw.de`

Abstract. Relational composition is an associative operation; therefore semigroup considerations often help in relational algebra. We study here some less known such effects and relate them with maximal rectangles inside a relation, i.e., with the basis of concept lattice considerations. The set of points contained in precisely one maximal rectangle makes up the fringe. We show that the converse of the fringe sometimes acts as a generalized inverse of a relation. Regular relations have a generalized inverse. They may be characterized by an algebraic condition.

1 Introduction

Relation algebra has had influx from semigroup theory, but only a study in a point-free form seems to offer chances to use it in a wider range. Inverses need not exist in general; the containment-ordering of relations, however, allows to consider sub-inverses. Occasionally the greatest sub-inverse also meets the requirements of an inverse. In interesting cases as they often originate from applications, not least around variants of orderings (semiorder, interval order, block-transitive order, e.g.), an inverse is needed and it may be characterized by appropriate means from that application area. It seems that this new approach generalizes earlier ones and at the same time facilitates them. In particular, semiorder considerations in [7] get a sound algebraic basis.

2 Prerequisites

We assume much of relation algebra to be known in the environment of RelMiCS, to be found not least in our standard reference [8,9], and concentrate on a few less known, unknown, or even new details. Already here, we announce two points: Unless explicitly stated otherwise, all our relations are possibly heterogeneous relations. When we quantify $\forall X, \exists X$, we always mean *... for which the construct in question is defined*.

A relation A is **difunctional**¹ if $A \cdot A^\top \cdot A \subseteq A$, which means that A can be written in block diagonal form by suitably rearranging rows and columns. If A is difunctional, the same obviously holds for $A^\top$.

¹ In [1] called a *matching relation* or simply a *match*.

If A, R are relations, f is a mapping, and x is a point, then negation commutes with composition so that $f:\overline{A} = \overline{f:A}$ as well as $\overline{R}:x = \overline{R:x}$.

Given any two relations R, S with coinciding domain, their **left residuum** is defined as $R \setminus S := \overline{R^\top : S}$, and correspondingly for P, Q with coinciding codomain their **right residuum** $Q / P := \overline{\overline{Q} : P^\top}$.

Combining this, we define the **symmetric quotient** $\text{syq}(A, B) := \overline{A^\top : B} \cap \overline{A^\top} : B$ for any two relations A, B with coinciding domain. Obviously, $\text{syq}(A, B) = A \setminus B \cap \overline{A} \setminus \overline{B}$. We recall several canceling formulae for the symmetric quotient: For arbitrary relations A, B, C we have

$$\begin{aligned} \text{syq}(A, B) : \text{syq}(B, C) &= \text{syq}(A, C) \cap \text{syq}(A, B) : \mathbb{T} \\ &= \text{syq}(A, C) \cap \mathbb{T} : \text{syq}(B, C) \subseteq \text{syq}(A, C). \end{aligned}$$

If $\text{syq}(A, B)$ is total, or if $\text{syq}(B, C)$ is surjective, then

$$\text{syq}(A, B) : \text{syq}(B, C) = \text{syq}(A, C).$$

For a given relation R , we define its corresponding **row-contains preorder**² $\mathcal{R}(R) := \overline{R} : R^\top = R / R$ and **column-is-contained preorder** $\mathcal{C}(R) := \overline{R^\top} : \overline{R} = R \setminus R$.

Given an ordering “ $\leq_E$ ”, resp. E , one traditionally calls the element $s \in V$ an **upper bound** of the set $U \subseteq V$ provided $\forall u \in U : u \leq_E s$. In point-free form we use the always existing — but possibly empty — set $\text{ubd}_E(U) = \overline{E^\top} : U$. Having this in mind, we introduce for any relation R two functionals, namely

$$\begin{aligned} \text{ubd}_R(X) &:= \overline{R^\top} : X, \text{ the upper bound cone functional and} \\ \text{lbd}_R(X) &:= \overline{\overline{R} : X}, \text{ the lower bound cone functional.} \end{aligned}$$

They are built in analogy to the construct given before, however, without assuming the relation R to be an ordering, nor need it be a homogeneous relation. The most important properties may nevertheless be shown using the Schröder equivalences.

2.1 Proposition. Given any fitting relations R, X , the following hold

- i) $\text{ubd}_R(\text{lbd}_R(\text{ubd}_R(X))) = \text{ubd}_R(X)$, i.e., $\overline{\overline{\overline{R^\top} : \overline{R} : \overline{R} : X}} = \overline{R^\top} : X$
- ii) $\text{lbd}_R(\text{ubd}_R(\text{lbd}_R(X))) = \text{lbd}_R(X)$, i.e., $\overline{\overline{\overline{\overline{R} : \overline{R} : \overline{R} : X}}} = \overline{\overline{R} : X}$ □

These formulae are really general, but have been studied mostly in more specialized contexts so far. We now get rid of any additional assumptions that are unnecessary and just tradition of the respective application field.

For the symmetric quotient, we once more refer to our standard reference [8,9] and add a new result here.

2.2 Proposition. For any fitting relations R, X, Y

$$\text{syq}(\text{lbd}_R(X), \text{lbd}_R(\text{ubd}_R(Y))) = \text{syq}(\text{ubd}_R(\text{lbd}_R(X)), \text{ubd}_R(Y)).$$

² In French: préordre finissant and préordre commençant; [5].

Proof: Applying $\text{syq}(A, B) = \text{syq}(\overline{A}, \overline{B})$ first, this expands to

$$\begin{aligned}\text{syq}(\overline{R}; X, \overline{R}; \overline{R}^\top; Y) &= \overline{\overline{X^\top; \overline{R}^\top; \overline{R}; \overline{R}^\top; Y}} \cap \overline{\overline{X^\top; \overline{R}^\top; \overline{R}; \overline{R}^\top; Y}} \\ \text{syq}(\overline{R}^\top; \overline{R}; X, \overline{R}^\top; Y) &= \overline{\overline{X^\top; \overline{R}^\top; \overline{R}; \overline{R}^\top; Y}} \cap \overline{\overline{X^\top; \overline{R}^\top; \overline{R}; \overline{R}^\top; Y}}\end{aligned}$$

Now, the first term in the first equals the second term in the second line. The other terms may be transformed into one another, applying Prop. 2.1. $\square$

With the symmetric quotient we may characterize membership relations ε , demanding $\text{syq}(\varepsilon, \varepsilon) \subseteq \mathbb{I}$ to hold as well as surjectivity $\text{syq}(\varepsilon, R)$ for arbitrary relations R . Using this, the containment ordering on the powerset may be built as $\Omega := \overline{\varepsilon^\top; \varepsilon} = \varepsilon \setminus \varepsilon$.

3 Rectangles

For an order, e.g., we observe that every element of the set u of elements smaller than some element e is related to every element of the set v of elements greater than e . Also for equivalences and preorders, square zones in the block-diagonal have proven to be important, accompanied by possibly rectangular zones off diagonal.

3.1 Definition. Given $u \subseteq X$ and $v \subseteq Y$, together with compatible universal relations $\mathbb{T}$, we call $u; v^\top = u; \mathbb{T} \cap (v; \mathbb{T})^\top$ a **rectangular relation** or, simply, a **rectangle**³. We say that u, v define a **rectangle inside** R if $u; v^\top \subseteq R$ (or equivalently $\overline{R}; v \subseteq \overline{u}$, or $\overline{R}^\top; u \subseteq \overline{v}$). $\square$

The definitional variants obviously mean the same. Sometimes we speak correspondingly of a **rectangle containing** R if $R \subseteq u; v^\top$, or we say that u, v is a **rectangle outside** R if u, v is a rectangle *inside* $\overline{R}$. Note that yet another definition of a rectangle u, v inside R may be given by $u \subseteq R/v^\top$ and $v^\top \subseteq u \setminus R$.

Although not many scientists seem to be aware of this fact, a significant amount of our reasoning is concerned with “rectangles” in/of a relation. A lower bound cone of an arbitrary relation R together with its upper bound cone form a rectangle inside R . Rectangles are handled at various places from the theoretical point of view as well as from the practical side. Among the application areas are concept lattices, clustering methods, and measuring, to mention just a few seemingly unrelated ones. In most cases, rectangles are treated in the respective application environment, i.e., together with certain additional properties, so that their status as rectangles is not clearly recognized, and consequently the corresponding algebraic properties are not applied or not fully exposed.

We now consider rectangles inside a relation that cannot be enlarged.

³ There are variant notations. In the context of bipartitioned graphs, a rectangle inside a relation is called a *block*; see, e.g. [3]. [4] speaks of *cross vectors*.

3.2 Definition. The rectangle u, v inside R is said to be **maximal**⁴ if for any rectangle u', v' inside R with $u \subseteq u'$ and $v \subseteq v'$, it follows that $u = u'$ and $v = v'$. $\square$

The property of being maximal has an elegant algebraic characterisation.

3.3 Proposition. Let u, v define a rectangle inside the relation⁵ R . Precisely when both, $\overline{R} \cdot v \supseteq \overline{u}$ and $\overline{R}^\top \cdot u \supseteq \overline{v}$, are also satisfied, there will not exist a strictly greater rectangle u', v' inside R .

Proof: Let us assume a rectangle that does not satisfy, e.g., the first inclusion: $\overline{u} \not\supseteq \overline{R} \cdot v$, so that there will exist a point $p \subseteq \overline{u} \cap \overline{R} \cdot v$. Then $u' := u \cup p \neq u$ and $v' := v$ is a strictly greater rectangle because $p \cdot v^\top \subseteq R$.

Consider for the opposite direction a rectangle u, v inside R satisfying the two inclusions together with a rectangle u', v' inside R such that $u \subseteq u'$ and $v \subseteq v'$. Then we may conclude with monotony and an application of the Schröder rule that $\overline{v'} \supseteq \overline{R}^\top \cdot u' \supseteq \overline{R}^\top \cdot u \supseteq \overline{v}$. This results in $v' = v$. In a similar way it is shown that $u = u'$. To sum up, u', v' can not be strictly greater than u, v . $\square$

In other words, u, v constitute a maximal rectangle inside R if and only if both, $\overline{R} \cdot v = \overline{u}$ and $\overline{R}^\top \cdot u = \overline{v}$, are satisfied. A reformulation of these conditions using residuals is $u = R/v^\top$ and $v^\top = u \setminus R$. Consider a pair of elements (x, y) related by some relation R , i.e., $x \cdot y^\top \subseteq R$ or, equivalently, $x \subseteq R \cdot y$. The relation $R^\top \cdot x$ is the set of all elements of the codomain side related with x . Since we started with $(x, y) \in R$, it is nonempty, i.e., $\perp \neq y \subseteq R^\top \cdot x$.

For reasons we will accept shortly, it is advisable to use the identity $R^\top \cdot x = \overline{\overline{R}^\top \cdot x}$ which holds because negation commutes with multiplying a point from the right side. We then see that a whole rectangle — may be only a one-element relation — is contained in R . Some preference has just been given to x , so that we expect something similar to hold when starting from y .

3.4 Proposition. Every point $x \cdot y^\top \subseteq R$ in a relation R gives rise to

i) the maximal rectangle inside R **started horizontally**

$$u_x := \overline{\overline{R} \cdot \overline{R}^\top \cdot x} = \overline{\overline{R} \cdot R^\top \cdot x} \supseteq x, \quad v_x := \overline{\overline{R}^\top \cdot x} = R^\top \cdot x \supseteq y$$

ii) the maximal rectangle inside R **started vertically**

$$u_y := \overline{\overline{R} \cdot y} = R \cdot y \supseteq x, \quad v_y := \overline{\overline{R}^\top \cdot \overline{R} \cdot y} = \overline{\overline{R}^\top \cdot R \cdot y} \supseteq y$$

Proof: Indeed, u_x, v_x as well as u_y, v_y are maximal rectangles inside R since they both satisfy Prop. 3.3. $\square$

These two may coincide, a case to be handled soon. One will find out that — although R has again not been defined as an ordering — the construct is similar to those defining upper bound sets and lower bound sets of upper bound sets.

⁴ In case, R is a homogeneous relation, it is also called a *diclique*, preferably with $u \neq \perp$ as well as $v \neq \perp$ to exclude trivialities; [3].

⁵ We assume a finite representable relation algebra satisfying the point axiom.

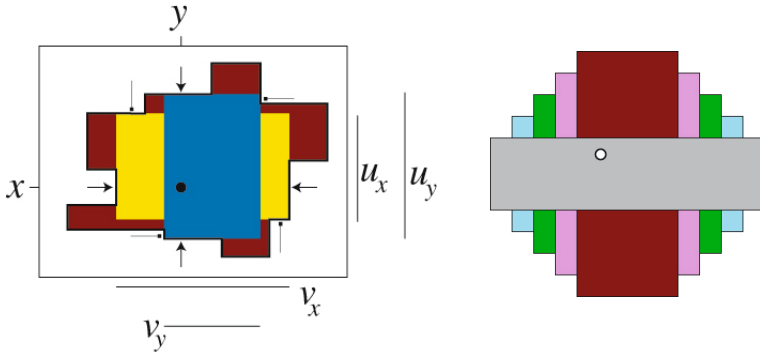


Fig. 1. Points contained in maximal rectangles

In Fig. 1, let the left relation R in question be the “non-white” area, inside which we consider an arbitrary pair (x, y) of elements related by R . To illustrate the pair (u_x, v_x) , let the point (x, y) first slide inside R horizontally over the maximum distance v_x , limited as indicated by $\rightarrow \leftarrow$. Then move the full subset v_x as far as possible inside R vertically, obtaining u_x , and thus, the light-shaded rectangle. Symbols like $\blacksquare$ indicate where the light grey-shaded rectangle cannot be enlarged in vertical direction.

In much the same way, slide the point (x, y) on column y as far as possible inside R , obtaining u_y , limited by $\downarrow$ and $\uparrow$. This vertical interval is then moved horizontally inside R as far as possible resulting in v_y and in the dark-shaded rectangle, confined by $\blacksquare$.

Observe, that the maximal rectangles need not be coherent in the general case; nor need there be just two. The example on the right of Fig. 1, where the relation considered is assumed to be precisely the union of all rectangles, shows a point contained in five maximal rectangles. What will also become clear is that with those obtained by looking for the maximum horizontal or vertical extensions first, one gets extreme cases.

As already announced, we now study the circumstances under which a point (x, y) is contained in exactly one maximal rectangle.

3.5 Proposition. A pair (x, y) of points related by R is contained in exactly one maximal rectangle inside R precisely when $x:y^\top \subseteq R \cap \overline{R:\overline{R}^\top}:\overline{R}$.

Proof: If there is just one maximal rectangle for $x:y^\top \subseteq R$, the extremal rectangles according to Prop. 3.4.i,ii will coincide. The proof then uses

$$\overline{R:\overline{R}^\top}:\overline{R} \supseteq \overline{R:y} \iff x:y^\top \subseteq \overline{R:\overline{R}^\top}:\overline{R} \quad \square$$

Important concepts concerning relations depend heavily on rectangles. For example, a decomposition into a set of maximal rectangles, or even dicliques, provides an efficient way of storing information in a database; see, e.g., [3].

3.6 Proposition. Given any relation R , the following constructs determine the set of all maximal rectangles — including the trivial ones with one side empty

and the other side full. Let ε be the membership relation starting from the domain side and ε' the corresponding one from the codomain side. Let Ω, Ω' be the corresponding powerset orderings. The construct

$$\begin{aligned} \Lambda &:= \text{syq}(\bar{\varepsilon}, \bar{R}; \varepsilon') \cap \text{syq}(\bar{R}^\top; \varepsilon, \bar{\varepsilon}') \text{ or, equivalently,} \\ \Lambda &:= \text{syq}(\varepsilon, \text{lbd}_R(\varepsilon')) \cap \text{syq}(\text{ubd}_R(\varepsilon'), \varepsilon') \end{aligned}$$

serves to relate $1 : 1$ the row sets to the column sets of the maximal rectangles.

Proof: Using $\varepsilon, \varepsilon'$, apply the condition Prop. 3.3 for a maximal rectangle simultaneously to all rows, or columns, respectively.

It is easy to convince oneself that Λ is a matching, i.e., satisfies $\Lambda^\top; \Lambda \subseteq \mathbb{I}$ and $\Lambda; \Lambda^\top \subseteq \mathbb{I}$. We show one of the cases using cancellation of the symmetric quotient together with the characterization of the membership relation ε' :

$$\begin{aligned} \Lambda^\top; \Lambda &= (\text{syq}(\bar{\varepsilon}, \bar{R}; \varepsilon') \cap \text{syq}(\bar{R}^\top; \varepsilon, \bar{\varepsilon}'))^\top; (\text{syq}(\bar{\varepsilon}, \bar{R}; \varepsilon') \cap \text{syq}(\bar{R}^\top; \varepsilon, \bar{\varepsilon}')) \\ &\subseteq \text{syq}(\bar{\varepsilon}', \bar{R}^\top; \varepsilon); \text{syq}(\bar{R}^\top; \varepsilon, \bar{\varepsilon}') \subseteq \text{syq}(\bar{\varepsilon}', \bar{\varepsilon}') = \text{syq}(\varepsilon', \varepsilon') = \mathbb{I} \quad \square \end{aligned}$$

Now we consider those rows/columns that participate in a maximal rectangle and extrude the respective rows/columns with ι to inject the subset described by the vector $\Lambda; \Pi$ and ι' to inject the subset described by the vector $\Lambda^\top; \Pi$. This allows us to define the two versions of the concept lattice based on the powerset orderings.

$$\text{left concept lattice} := \iota; \Omega; \iota^\top \quad \text{right concept lattice} := \iota'; \Omega'; \iota'^\top.$$

The two, sometimes referred to as lattice of *extent*, or *intent* resp., are $1 : 1$ related by the matching $\lambda := \iota; \Lambda; \iota'^\top$.

4 Fringes

The points contained in just one maximal rectangle inside a relation R play an important rôle, so that we introduce a notation for them.

4.1 Definition. For arbitrary R we define its **fringe**(R) := $R \cap \overline{R; \bar{R}^\top}; R$. $\square$

A first inspection shows that **fringe**($R^\top$) = $[\text{fringe}(R)]^\top$. The concept of a fringe has unexpectedly many applications. We announce already here that every fringe will turn out to be difunctional, and thus enjoys a powerful “geometric characterization as a (possibly partial) block-diagonal”. As a first example for this, we mention that the fringe of an ordering E is the identity, since

$$\text{fringe}(E) = E \cap \overline{E; \bar{E}^\top}; E = E \cap \overline{\bar{E}^\top}; E = E \cap \overline{\bar{E}}^\top = E \cap E^\top = \mathbb{I}.$$

We are accustomed to use the identity $\mathbb{I}$. For heterogeneous relations there is none; often in such cases, the fringe takes over and may be made similar use of.

The fringe of the strict order C is always contained in its Hasse relation $H := C \cap \overline{C^2}$ since C is irreflexive. The existence of a non-empty fringe heavily depends on finiteness or at least discreteness. The following resembles a result of Michael Winter [10]. Let us for a moment call C a **dense** relation if it satisfies

$C:C = C$. An example is obviously the relation “ $<$ ” on the real numbers. This strict order is transitive $C:C \subseteq C$, but satisfies also $C \subseteq C:C$, meaning that whatever element relationship one chooses, e.g., $3.7 < 3.8$, one will find an element in between, $3.7 < 3.75 < 3.8$. To be a dense relation implies that the Hasse relation will be empty. A dense linear strict ordering has an empty fringe. We show in the subsequent sections that the fringe of a relation is central for difunctional, Ferrers, and block-transitive relations.

Now we present a plexus of formulae that are heavily interrelated. The fringe gives rise to “partial equivalences” or symmetric idempotents, closely resembling row and column equivalence $\Xi(R) := \text{syq}(R^\top, R^\top) = \text{syq}(\overline{R:R^\top}^\top, \overline{R:R^\top}^\top)$ and $\Psi(R) := \text{syq}(R, R) = \text{syq}(\overline{R^\top:R}^\top, \overline{R^\top:R}^\top)$.

4.2 Definition. For an arbitrary relation R and its fringe $f := \text{fringe}(R) = R \cap \overline{R:R^\top:R}$, we define

- i) $\Xi_F(R) := f:f^\top$, the **fringe-partial row equivalence**
- ii) $\Psi_F(R) := f^\top:f$, the **fringe-partial column equivalence** □

We recall that the fringe collects those entries of a relation R that are contained in precisely one maximal rectangle. The fringe may also be obtained with the symmetric quotient from the row-contains-preorder and the relation in question:

4.3 Proposition. For an arbitrary relation R , the fringe and the row-contains-preorder $\mathcal{R}(R)$, satisfy

$$\text{fringe}(R) = \text{syq}(\mathcal{R}(R), R)$$

Proof: We expand **fringe**, **syq**, $\mathcal{R}(R)$, and apply trivial operations to obtain

$$R \cap \overline{R:R^\top:R} = \overline{R:R^\top:R} \cap \overline{\overline{R:R^\top:R}^\top:R}$$

It remains, thus, to finally apply that $\overline{R} = \overline{R:R^\top:R}$. □

Thus, we are allowed to make use of cancellation formulae from Sect. 2 for the symmetric quotient. We show that to a certain extent the row equivalence $\Xi(R)$ may be substituted by $\Xi_F(R)$; both coincide as long as the fringe is total. They may be different, but only in the way that a square diagonal block of the **fringe-partial row equivalence** is either equal to the one in $\Xi(R)$, or empty.

4.4 Proposition. For an arbitrary relation R and its fringe $f := \text{fringe}(R)$ the **fringe-partial row resp. column equivalences** satisfy the following:

- i) $\Xi_F(R) = \Xi(R) \cap f:\Pi$
- ii) $\Xi(R):f = \Xi_F(R):f = f:f^\top:f = f \quad f = f:f^\top:f = f:\Psi_F(R) = f:\Psi(R)$
- iii) $f^\top:\Xi(R):f \subseteq \Psi(R)$
- iv) $\Xi_F(R):R \subseteq R:f^\top:R \subseteq R \quad \text{and} \quad R:\Psi_F(R) \subseteq R:f^\top:R \subseteq R$.

- Proof:** i) $\Xi_F(R) = f; f^\top$ Def. 4.2
- $$= \text{syq}(\overline{R; R^\top}, R); \text{syq}(R, \overline{R; R^\top})$$
- Prop. 4.3
- $$= \text{syq}(\overline{R; R^\top}, \overline{R; R^\top}) \cap \text{syq}(\overline{R; R^\top}, R); \Pi$$
- cancellation property
- $$= \Xi(\overline{R}) \cap f; \Pi = \Xi(R) \cap f; \Pi$$
- definition of
- $\Xi(R)$
- ii) The definition of $\Xi(R)$ together with Prop. 4.3 show that
- $$\Xi(R); f = \Xi(\overline{R}); f = \text{syq}(\overline{R; R^\top}, \overline{R; R^\top}); \text{syq}(\overline{R; R^\top}, R) \subseteq \text{syq}(\overline{R; R^\top}, R) = f,$$
- applying cancellation again. Then we may proceed with $f; f^\top; f = \Xi_F(R); f \subseteq$
- $$\subseteq \Xi(R); f$$
- according to (i)
- $$\subseteq f$$
- see above
- $$\subseteq f; f^\top; f$$
- $A \subseteq A; A^\top; A$
- for every
- A
- obtaining equality everywhere in between.
- iii) $f^\top; \Xi(R); f \subseteq f^\top; f$ see above
- $$= \Psi_F(R)$$
- by Def. 4.2
- $$\subseteq \Psi(R)$$
- applying (i) to
- $R^\top$
- .
- iv) $R; f^\top; R$
- $$= R; [\text{syq}(\overline{R; R^\top}, R)]^\top; R$$
- Prop. 4.3
- $$= R; \text{syq}(R, \overline{R; R^\top}); R$$
- transposing a symmetric quotient
- $$\subseteq \overline{R; R^\top}; R$$
- cancelling the symmetric quotient
- $$\subseteq R$$
- which holds for every relation

The rest is then simple because $\Xi_F(R) = f; f^\top \subseteq R; f^\top$. $\square$

Anticipating Def. 5.1, we may say that $f^\top$ is always a subinverse of R . We have already seen in (i) that $\Xi_F(R)$ is nearly an equivalence. When in (iv) equality holds, $\Xi_F(R); R = R$, we may expect important consequences, since then something as a congruence is established.

The following proposition relates the fringe of the row-contains-preorder with the row equivalence.

4.5 Proposition. We have for every relation R , that

$$\text{fringe}(\mathcal{R}(R)) = \text{fringe}(\overline{R; R^\top}) = \text{syq}(R^\top, R^\top) = \Xi(R),$$

$$\text{fringe}(\mathcal{C}(R)) = \text{fringe}(\overline{R^\top}; R) = \text{syq}(R, R) = \Psi(R).$$

Proof: In both cases, only the equality in the middle is important because the rest is just expansion of definitions. Thus reduced, the first identity, e.g., requires to prove that

$$\overline{R; R^\top} \cap \overline{\overline{R; R^\top}; R; \overline{R; R^\top}} = \overline{R; R^\top} \cap \overline{R; \overline{R; R^\top}}.$$

The first term on the left equals the first on the right. In addition, the second terms are equal, which is not seen so easily, but also trivial. $\square$

The fringe may indeed be important because it is intimately related with difunctionality: For arbitrary R , the construct $\text{fringe}(R)$ is difunctional and a relation R is difunctional precisely when $R = \text{fringe}(R)$. Also: Forming the fringe turns out to be an idempotent operation, i.e., $\text{fringe}(\text{fringe}(R)) = \text{fringe}(R)$.

5 Inverses

Fringes and difunctionality are related to the following concepts of inverses. Inverses are defined for real-valued matrices in linear algebra or for numerical problems. We introduce here similar definitions for relations using the same names. They will provide deeper insight into the structure of a difunctional relation.

5.1 Definition. Let some relation A be given. The relation G is called

- i) a **sub-inverse** of A if

$$A \circ G \circ A \subseteq A.$$
- ii) a **generalized inverse** of A if

$$A \circ G \circ A = A.$$
- iii) a **Thierrin-Vagner inverse** of A if the following two conditions hold

$$A \circ G \circ A = A, \quad G \circ A \circ G = G.$$
- iv) a **Moore-Penrose inverse** of A if the following four conditions hold

$$A \circ G \circ A = A, \quad G \circ A \circ G = G, \quad (A \circ G)^\top = A \circ G, \quad (G \circ A)^\top = G \circ A.$$

The relation R is called **regular**, if it has a generalized inverse. Due to the symmetric situation in case of a Thierrin-Vagner inverse G of A , the two relations A, G are also simply called **inverses** of each other. $\square$

In a number of situations semigroup theory is applicable to relations. Some of these ideas stem from [4] and are here reconsidered from the relational side. A sub-inverse will always exist since $\mathbb{I}$ satisfies the requirement. With two sub-inverses G, G' also their union $G \cup G'$ is obviously a sub-inverse so that one will ask which is the greatest.

5.2 Proposition. $\overline{R \circ \overline{R}^\top \circ R}^\top$ is the greatest subinverse of R .

Proof: Assuming an arbitrary sub-inverse X of R , it satisfies by definition $R \circ X \circ R \subseteq R$, which is equivalent with

$$\iff X^\top \circ R^\top \circ \overline{R} \subseteq \overline{R} \iff R \circ \overline{R}^\top \circ R \subseteq \overline{X}^\top \iff X \subseteq \overline{R \circ \overline{R}^\top \circ R}^\top \quad \square$$

A generalized inverse is not uniquely determined: As an example assume a homogeneous $\mathbb{T}$. It has at least the generalized inverses $\mathbb{I}$ and $\mathbb{T}$. With generalized inverses G_1, G_2 also $G_1 \cup G_2$ is a generalized inverse. There will, thus, exist a greatest one — if any. Regular relations, i.e., those with existing generalized inverse, may precisely be characterized by the following containment which is in fact an equation:

5.3 Proposition. R regular $\iff R \subseteq \overline{R \circ \overline{R}^\top \circ R}^\top$.

Proof: If R is regular, there exists an X with $R \circ X \circ R = R$. It is, therefore, a sub-inverse and so $X \subseteq \overline{R \circ \overline{R}^\top \circ R}^\top$ according to Prop. 5.2. Then

$$R = R \circ X \circ R \subseteq R \circ \overline{R \circ \overline{R}^\top \circ R}^\top \circ R.$$

Specializing $X := \overline{R; \overline{R}^\top; R}^\top$ in the proof of Prop. 5.2, we have already seen that $R; \overline{R; \overline{R}^\top; R}^\top; R \subseteq R$ for arbitrary R . $\square$

We will learn in Def. 7.1, that every block-transitive relation is regular in this sense; see Prop. 7.3.

5.4 Proposition. If R is a regular relation, its maximum Thierrin-Vagner inverse is $\overline{R; \overline{R}^\top; R}^\top; \overline{R; \overline{R}^\top; R}^\top =: TV$.

Proof: Evaluation of $TV; R; TV = TV$ and $R; TV; R = R$ using Prop. 5.3 with equality shows that TV is indeed a Thierrin-Vagner inverse. Any Thierrin-Vagner inverse G is in particular a sub-inverse, so that $G \subseteq \overline{R; \overline{R}^\top; R}^\top$ which implies $G = G; R; G \subseteq TV$. $\square$

A well-known result on Moore-Penrose inverses shall be recalled:

5.5 Theorem. Moore-Penrose inverses are uniquely determined if they exist.

Proof: Assume two Moore-Penrose inverses G, H of A to be given. Then we may proceed as follows: $G = G; A; G = G; G^\top; A^\top = G; G^\top; A^\top; H^\top; A^\top = G; G^\top; A^\top; A; H = G; A; G; A; H = G; A; H = G; A; H; A; H = G; A; A^\top; H^\top; H = A^\top; G^\top; A^\top; H^\top; H = A^\top; H^\top; H = H; A; H = H$. $\square$

These concepts will now be related with permutations and difunctionality.

5.6 Theorem. For a relation A , the following are equivalent:

- i) A has a Moore-Penrose inverse.
- ii) A has $A^\top$ as its Moore-Penrose inverse.
- iii) A is difunctional.
- iv) Any two rows (or columns) of A are either disjoint or identical.
- v) There exist permutations P, Q such that $P; A; Q$ has block-diagonal form with (not necessarily square) diagonal entries $B_i = \mathbb{I}$.

Proof: of the key step (i) $\implies$ (ii): $G = G; A; G \subseteq G; A; A^\top; A; G = A^\top; G^\top; A^\top; A; G = (A; G; A)^\top; A; G = A^\top; A; G = A^\top; G^\top; A^\top = (A; G; A)^\top = A^\top$ and, deduced symmetrically, $A \subseteq G^\top$. $\square$

6 Ferrers Relations

We have seen that a difunctional relation corresponds to a partial block diagonal relation. So the question arose as to whether there was a counterpart of a linear order with rectangular block-shaped matrices. In this context, the Ferrers property of a relation is studied.

6.1 Definition. We say that a relation A is **Ferrers** if $A; \overline{A}^\top; A \subseteq A$. $\square$

The meaning of the algebraic condition has often been visualized and interpreted. It is at first sight not at all clear that the matrix representing A may — due to Ferrers property — be written in staircase (or echelon) block form after suitably rearranging rows and columns independently.

If R is Ferrers, then so are $R^\top$, $\overline{R}$, $\overline{R}^\top; R$, $\overline{R}; R^\top$, and $R; \overline{R}^\top; R$. A relation R is Ferrers precisely when $\mathcal{R}(R)$ is connex or when $\mathcal{C}(R)$ is connex⁶.

$$R; \overline{R}^\top; R \subseteq R \iff \overline{R}; R^\top; \overline{R} \subseteq \overline{R} \iff R; \overline{R}^\top \subseteq \overline{R}; R^\top$$

We now prove several properties of a Ferrers relation that make it attractive for purposes of modeling preferences etc. An important contribution to this comes from a detailed study of the behaviour of the fringe⁷.

6.2 Proposition. For a finite Ferrers relation R , the following statements hold, in which we abbreviate $f := \text{fringe}(R)$:

- i) The construct $R; \overline{R}^\top$ is a progressively bounded semi-connex strict order.
- ii) There exists a natural number $k \geq 0$ that gives rise to a strictly increasing exhaustion as

$$\mathbb{1} = (R; \overline{R}^\top)^k \subsetneq (R; \overline{R}^\top)^{k-1} \subsetneq \dots \subsetneq R; \overline{R}^\top; R; \overline{R}^\top \subsetneq R; \overline{R}^\top$$

$$\text{iii) } R; \overline{R}^\top = f; \overline{R}^\top, \quad \overline{R}^\top; R = \overline{R}^\top; f, \quad R; \overline{R}^\top; R = f; \overline{R}^\top; f$$

- iv) R allows a disjoint decomposition as

$$R = \text{fringe}(R) \cup \text{fringe}(R; \overline{R}^\top; R) \cup \dots \cup \text{fringe}((R; \overline{R}^\top)^k; R) \text{ for some } k \geq 0$$

- v) R allows a disjoint decomposition as

$$R = \text{fringe}(R) \cup \text{fringe}(f; \overline{R}^\top; f) \cup \dots \cup \text{fringe}((f; \overline{R}^\top)^k; f) \text{ for some } k \geq 0$$

- vi) R allows a disjoint decomposition as $R = f \cup f; \overline{R}^\top; f$

- vii) R allows an exhaustion as

$$\mathbb{1} = (f; \overline{R}^\top)^k; f \subsetneq (f; \overline{R}^\top)^{k-1}; f \subsetneq \dots \subsetneq f; \overline{R}^\top; f; \overline{R}^\top; f \subsetneq f; \overline{R}^\top; f \subsetneq R$$

Proof: i) and ii) We start the following chain of inclusions from the right applying recursively that R is Ferrers:

$$\mathbb{1} = (R; \overline{R}^\top)^k \subseteq (R; \overline{R}^\top)^{k-1} \subseteq \dots \subseteq R; \overline{R}^\top; R; \overline{R}^\top \subseteq R; \overline{R}^\top$$

Finiteness guarantees that it will eventually be stationary, i.e., $(R; \overline{R}^\top)^{k+1} = (R; \overline{R}^\top)^k$. This means in particular that the condition $Y \subseteq (R; \overline{R}^\top); Y$ holds for $Y := (R; \overline{R}^\top)^k$. The construct $R; \overline{R}^\top$ is obviously transitive and irreflexive, so that it is in combination with finiteness also progressively finite. According to Sect. 6.3 of [8,9], this means that $Y = (R; \overline{R}^\top)^k = \mathbb{1}$.

$$\begin{aligned} \text{iii) } R; \overline{R}^\top &= ((R \cap \overline{R; \overline{R}^\top; R}) \cup (R \cap R; \overline{R}^\top; R)); \overline{R}^\top \\ &= (f \cup R; \overline{R}^\top; R); \overline{R}^\top \end{aligned}$$

since R is Ferrers

⁶ A relation A is connex if $\Pi = A \cup A^\top$; it is semi-connex if $\overline{\Pi} \subseteq A \cup A^\top$.

⁷ By the way: [6] and a whole chapter of [7] are devoted to the “holes” or “hollows” and “noses” that show up in this context; see Fig. 2.

$$\begin{aligned}
&= f; \overline{R}^\top \cup R; \overline{R}^\top; R; \overline{R}^\top \\
&= f; \overline{R}^\top \cup (f; \overline{R}^\top \cup R; \overline{R}^\top; R; \overline{R}^\top); R; \overline{R}^\top && \text{applied recursively} \\
&= f; \overline{R}^\top \cup f; \overline{R}^\top; R; \overline{R}^\top \cup R; \overline{R}^\top; R; \overline{R}^\top; R; \overline{R}^\top \\
&= f; \overline{R}^\top \cup R; \overline{R}^\top; R; \overline{R}^\top; R; \overline{R}^\top && \text{since also } \overline{R}^\top \text{ is Ferrers} \\
&= \dots = f; \overline{R}^\top \cup \mathbb{1} && \text{see (ii)}
\end{aligned}$$

The other proofs are left to the reader. $\square$

It is mainly this effect which enables us to arrive at the results that follow. First, we observe how successively discarding fringes leaves a decreasing sequence of relations; strictly decreasing when finite or at least not dense. Ferrers relations may, although possibly heterogeneous, in many respects be considered as similar to a linear (strict) ordering.

The following proposition is a classic (with a very slight generalization concerning surjectivity not being demanded); it may not least be found in [2] and also with a completely different point-free proof in [9]. The idea of the proof presented here is a constructive one, which means that one may write the constructs down in the language **TITUREL** and immediately run this as a program. The reason is that the constructs are generic ones that are uniquely characterized, so that a standard realization for interpretation is possible.

6.3 Proposition. Let $R : X \longrightarrow Y$ be a finite relation.

$$R \text{ Ferrers} \iff \text{There exist mappings } f, g \text{ and a linear strict order } C \text{ such that } R = f; C; g^\top.$$

Proof: “ $\Leftarrow$ ” follows relatively easily using several times that mappings may slip below a negation from the left without affecting the result, and that C is Ferrers.

“ $\Rightarrow$ ” Let R be Ferrers. There may exist empty rows or columns in R or not. To care for this in a general form, we enlarge the domain to $X + \mathbb{1}$ and the codomain to $\mathbb{1} + Y$ and consider the relation $R' := \iota_X^\top; R; \kappa_Y$. In R' , there will definitely exist at least one empty row and at least one empty column. It is intuitively clear — and easy to demonstrate — that also R' is Ferrers.

The relation R' has been constructed so that $\overline{R}'$ is both, total and surjective. Observe, that $\overline{R}$ in the upper right sub-rectangle of Fig. 2 would *not* have been surjective. As in general $R = f \cup f; \overline{R}^\top; f$ according to Prop. 6.2.vi, also $\text{fringe}(\overline{R}')$ is necessarily total and surjective. As fringes are always difunctional, $\text{fringe}(\overline{R}')$ is a block diagonal, which will — after quotient forming — provide us with the matching λ .

We introduce row equivalence $\Xi(R') := \text{syq}(R'^\top, R')$ as well as column equivalence $\Psi(R') := \text{syq}(R', R')$ of R' together with the corresponding natural projections which we call η_Ξ, η_Ψ . We define

$$\begin{aligned}
\lambda &:= \eta_\Xi^\top; \text{fringe}(\overline{R}'); \eta_\Psi \\
f &:= \iota_X; \eta_\Xi; \lambda \\
g &:= \kappa_Y; \eta_\Psi \\
C &:= \lambda^\top; \eta_\Xi^\top; R'; \eta_\Psi
\end{aligned}$$

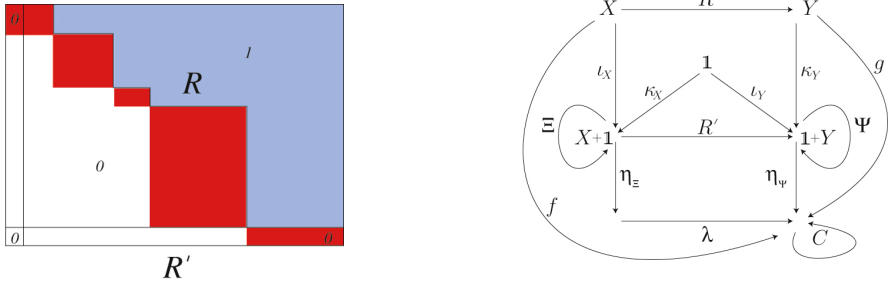


Fig. 2. Constructing a Ferrers decomposition

Now a proof is achievable requiring no case distinctions which are impossible prior to having interpreted the relation in question “with a matrix”. $\square$

7 Block-Transitive Relations

Concepts that we already know for an order or a strict order shall now be studied generalized to a heterogeneous environment in which also multiple rows or columns may occur. The starting point is a Ferrers relation. We have seen how it can in many respects be compared with a linear (strict) order. Is it possible to obtain in such a generalized case similar results for a not necessarily linear strict order? Proceeding strictly algebraically, this will indeed be found.

7.1 Definition. A relation R is called **block-transitive** if either one of the following equivalent conditions holds, expressed via its fringe $f := \text{fringe}(R)$

- i) $R \subseteq f; \mathbb{T}$ and $R \subseteq \mathbb{T}; f$,
- ii) $R \subseteq f; \mathbb{T}; f$,
- iii) $R = \Xi_F; R; \Psi_F$.

$\square$

The proof of the equivalence of the variants is left to the reader. Being block-transitive is mainly a question of how big the fringe is. The fringe must be big enough so as to “span” the given relation R with its rectangular closure.

For this concept, Michael Winter had originally, see [10], coined the property to be of *order-shape*. We do not use this word here because it may cause misunderstanding: We had always been careful to distinguish an order from a strict order; they have different definitions, that both overlap in being transitive. In what follows, we will see that — in a less consistent way — definitions may share the property of being block-transitive.

The following shows the most specialized examples of a block-transitive relation:

7.2 Proposition. A difunctional relation R as well as a finite Ferrers relation R are necessarily block-transitive.

Proof: The first result is trivial since $R \cdot R^\top \cdot R \subseteq R \iff R \cdot \overline{R}^\top \cdot R \subseteq \overline{R}$, so that $R = \mathbf{fringe}(R)$. For the second, we abbreviate $f := \mathbf{fringe}(R)$. According to Prop. 6.2.iv, we have $R = f \cup f \cdot \overline{R}^\top \cdot f$, so that $R \subseteq f \cdot \mathbb{T}$ as well as $R \subseteq \mathbb{T} \cdot f$. $\square$

This is in contrast to $\mathbb{R}, <$ which is Ferrers but not block-transitive, simply since its fringe has already been shown to be empty.

7.3 Proposition. For an arbitrary block-transitive relation R we again abbreviate $f := \mathbf{fringe}(R)$ and prove:

- i) $R \cdot f^\top \cdot R = R$, i.e., $f^\top$ is a generalized inverse of R
- ii) $R \cdot f^\top$ and $f^\top \cdot R$ are transitive

Proof: i) From $R \subseteq f \cdot \mathbb{T}$, we deduce with row equivalence Ξ and Prop. 4.4.i

$$R = R \cap f \cdot \mathbb{T} = \Xi \cdot R \cap f \cdot \mathbb{T} = (\Xi \cap f \cdot \mathbb{T}) \cdot R = \Xi_F \cdot R = f \cdot f^\top \cdot R \subseteq R \cdot f^\top \cdot R$$

The reverse direction is satisfied for every relation according to Prop. 4.4.iv.

- ii) $R \cdot f^\top \cdot R \cdot f^\top = R \cdot f^\top$ using (i)

$\square$

We now introduce block-transitive kernels and Ferrers closures.

7.4 Definition. For a relation R , we define using its fringe f the **block-transitive kernel** as

$$\mathbf{btk}(R) := R \cap f \cdot \mathbb{T} \cap \mathbb{T} \cdot f = f \cdot f^\top \cdot R \cdot f^\top \cdot f. \quad \square$$

7.5 Proposition. For every relation R , the fringe does not change when reducing R to its block-transitive kernel; i.e.,

$$f = \mathbf{fringe}(R \cap f \cdot \mathbb{T} \cap \mathbb{T} \cdot f) \quad \text{for } f := \mathbf{fringe}(R) \quad \square$$

The proof of this statement is too lengthy and ugly to be presented. It employs hardly more than Boolean algebra, but with terms running in opposite directions, so that it is probably not easy for the reader to find it for himself.

7.6 Proposition. Every finite block-transitive relation R has a **Ferrers closure**, i.e., a Ferrers relation $F \supseteq R$ but still satisfying $\mathbf{fringe}(F) = \mathbf{fringe}(R)$.

Proof: The idea for this proof is rather immediate; its execution, though, is technically complicated: Do the quotient forming according to the fringe-partial equivalences throwing rows and columns with empty row/column of f together in one class. Divide these congruences out and apply afterwards what is called the Szpilrajn-extension or topological sorting. $\square$

For block-transitive relations, also a factorization result similar to Prop. 6.3 may be proved, which cannot be presented for reasons of space.

8 Concluding Remark

We have tried to base known and new concepts on maximal rectangles inside a relation. The elegant relational characterization of these together with the

intuitive interpretation of a fringe facilitated access to semigroup concepts, e.g., and also allowed to generalize some. Block-transitive relations constitute a novel concept that may turn out to be the method of choice in preference modeling. They are more general than semiorders or interval orders, but still allow an algebraic treatment.

References

1. Doignon, J.-P., Falmagne, J.-C.: Matching Relations and the Dimensional Structure of Social Sciences. *Math. Soc. Sciences* 7, 211–229 (1984)
2. Ducamp, A., Falmagne, J.-C.: Composite Measurement. *J. Math. Psychology* 6, 359–390 (1969)
3. Haralick, R.M.: The diclique representation and decomposition of binary relations. *J. ACM* 21, 356–366 (1974)
4. Kim, K.H.: *Boolean Matrix Theory and Applications*. Monographs and Textbooks in Pure and Applied Mathematics, vol. 70. Marcel Dekker, New York – Basel (1982)
5. Monjardet, B.: Axiomatiques et propriétés des quasi-ordres. *Mathematiques et Sciences Humaines* 16(63), 51–82 (1978)
6. Pirlot, M.: Synthetic description of a semiorder. *Discrete Appl. Mathematics* 31, 299–308 (1991)
7. Pirlot, M., Vincke, P.: *Semiorders — Properties, Representations, Applications*. Theory and Decision Library, Mathematical and Statistical Methods, Series B, vol. 36. Kluwer Academic Publishers, Dordrecht (1997)
8. Schmidt, G., Ströhlein, T.: *Relationen und Graphen*. Mathematik für Informatiker. Springer, Heidelberg (1989)
9. Schmidt, G., Ströhlein, T.: *Relations and Graphs — Discrete Mathematics for Computer Scientists*. In: *EATCS Monographs on Theoretical Computer Science*. Springer, Heidelberg (1993)
10. Winter, M.: Decomposing Relations Into Orderings. In: Berghammer, R., Möller, B., Struth, G. (eds.) *RelMiCS/AKA 2003*. LNCS, vol. 3051, pp. 261–272. Springer, Heidelberg (2004)

An Ordered Category of Processes

Michael Winter*

Department of Computer Science,
Brock University,
St. Catharines, Ontario, Canada, L2S 3A1
`mwinter@brocku.ca`

Abstract. Processes can be seen as relations extended in time. In this paper we want to investigate this observation by deriving an ordered category of processes. We model processes as co-algebras of a relator on Dedekind category up to bisimilarity. On those equivalence classes we define a lower semi-lattice structure and a monotone composition operation.

1 Introduction

With this paper we want to start a comprehensive study of processes seen as relations extended in time. At a given time step a process may perform an input/output action. This corresponds to a functional or, more general, a relational behavior. After performing the action the process may switch to another internal state, i.e. may become another process. This observation justifies that process can be modelled in non-well-founded set theories using fixed points of the power set functor [2]. A categorical investigation using this approach led to notion of interaction categories [1]. Interaction categories lack an allegorical structure, and, therefore, just cover some aspects of relations extended in times. The order structure, and, hence, the allegorical structure that is usually available for relation is ignored. Another approach introduced time-extended allegories [10]. This kind of structure provides all relational and, in addition, time related operations. Unfortunately, time-extended allegories do not have obvious models. In this paper we model processes as co-algebras of a relator on Dedekind category up to bisimilarity. We show that there is a natural ordering on the corresponding equivalence classes. Furthermore, we are going to define a notion of composition of processes based on parallel composition. We show that the structure we obtain is an ordered category.

2 Dedekind Categories

Throughout this paper, we use the following notation. To indicate that a morphism R of a category $\mathcal{R}$ has source A and target B we write $R : A \rightarrow B$. The

* The author gratefully acknowledges support from the Natural Sciences and Engineering Research Council of Canada.

collection of all morphisms $R : A \rightarrow B$ is denoted by $\mathcal{R}[A, B]$ and the composition of a morphism $R : A \rightarrow B$ followed by a morphism $S : B \rightarrow C$ by $R; S$. Last but not least, the identity morphism on A is denoted by $\mathbb{I}_A$.

In this section we recall some fundamentals on Dedekind categories [6, 7]. This kind of category is called locally complete division allegories in [4].

Definition 1. *A Dedekind category $\mathcal{R}$ is a category satisfying the following:*

1. *For all objects A and B the collection $\mathcal{R}[A, B]$ of morphisms/relations is a complete distributive lattice. Meet, join, the induced ordering, the least and the greatest element are denoted by $\sqcap, \sqcup, \sqsubseteq, \sqll_{AB}$ and $\sqtop_{AB}$, respectively.*
2. *There is a monotone operation $\smile$ (called converse) mapping a relation $Q : A \rightarrow B$ to a relation $Q^\smile : B \rightarrow A$ such that $(Q; R)^\smile = R^\smile; Q^\smile$ and $(Q^\smile)^\smile = Q$ for all relations $Q : A \rightarrow B$ and $R : B \rightarrow C$.*
3. *For all relations $Q : A \rightarrow B, R : B \rightarrow C$ and $S : A \rightarrow C$ the modular law $Q; R \sqcap S \sqsubseteq Q; (R \sqcap Q^\smile; S)$ holds.¹*
4. *For all relations $R : B \rightarrow C$ and $S : A \rightarrow C$ there is a relation $S/R : A \rightarrow B$ (called the left residual of S and R) such that for all $Q : A \rightarrow B$ the following holds: $Q; R \sqsubseteq S \iff Q \sqsubseteq S/R$.*

We will also need the weaker notion of an ordered category in this paper. An ordered category $\mathcal{C}$ is a category so that every collection $\mathcal{C}[A, B]$ is an ordered class and composition is a monotone operation. $\mathcal{C}$ is called an ordered category with converse iff it is an ordered category with an converse operation satisfying 2. of the previous definition. Notice that a Dedekind category is an ordered category with converse.

Throughout this paper we will use several basic properties of Dedekind categories such as $\mathbb{I}_A^\smile = \mathbb{I}_A$, the monotonicity of composition in both parameters or the distributivity of $;$ over $\sqcup$ without mentioning. For details we refer to any of [4, 8, 9, 10].

For relations $Q : A \rightarrow B$ and $R : A \rightarrow C$ one may define a right residual $Q \backslash R : B \rightarrow C$ by $Q \backslash R := (R^\smile / Q^\smile)^\smile$. This construction is characterized by $X \sqsubseteq Q \backslash R$ iff $Q; X \sqsubseteq R$.

As a description of recursive data types as well as labelled transition systems a special class of functors is of interest [3].

Definition 2. *Let F be a functor between ordered categories with converse. We call F a relator iff F is monotonic and preserves converse, i.e. $R \sqsubseteq S$ implies $F(R) \sqsubseteq F(S)$ and $F(R^\smile) = F(R)^\smile$.*

Notice, that the definition in [3] is slightly different. The allegories they consider are tabular (see below) so that the preservation of converse is equivalent to monotonicity. The notion of a relator was introduced by Y. Kawahara in [5].

Recall that a natural transformation $\eta : F \rightarrow G$ between two functors is a family of morphisms so that $F(f); \eta = \eta; G(f)$ for all suitable f . In the context of

¹ By convention the precedence of the operations decreases in the following order $\smile, ;, \sqcap$.

Dedekind categories and relators F and G one is often interested in lax natural transformations satisfying the weaker property $F(Q); \eta \sqsubseteq \eta; F(Q)$.

An important class of relations is given by mappings.

Definition 3. Let $Q : A \rightarrow B$ be a relation. Then we call

1. Q univalent iff $Q^\sim; Q \sqsubseteq \mathbb{I}_B$,
2. Q total iff $\mathbb{I}_A \sqsubseteq Q; Q^\sim$,
3. Q a map iff Q is univalent and total,
4. Q injective iff $Q^\sim$ is univalent,
5. Q surjective iff $Q^\sim$ is total.

Notice, that if Q is a bijective mapping, i.e. a mapping that is injective and surjective, we have $Q^\sim; Q = \mathbb{I}_B$ and $Q; Q^\sim = \mathbb{I}_A$.

A relator preserves all notions of the last definition. In particular, its restriction to the class of mappings is a functor between the corresponding subcategories.

In the next lemma we collect some fundamental facts used in this paper. A proof may be found in [48, 49].

Lemma 1. Let $Q, T : A \rightarrow B$, $R : B \rightarrow C$, $S : A \rightarrow D$, $V : B \rightarrow A$, and $U, Z : A \rightarrow C$. Then we have

1. $Q; R \sqcap S; \Pi_{DC} = (Q \sqcap S; \Pi_{DB}); R$ and $(Q; R \sqcap U); \Pi_{CD} = (Q \sqcap U; R^\sim); \Pi_{BD}$,
2. If U is univalent, then $U; (Q \sqcap T) = U; Q \sqcap U; T$,
3. If U is univalent, then $(V \sqcap R; U^\sim); U = V; U \sqcap R$,
4. If $X \sqsubseteq \mathbb{I}_A$, then $X; X = X$ and $X; (Q \sqcap T) = Q \sqcap X; T$,
5. If U is univalent, then $U \sqcap (U \sqcap Z); \Pi_{CA} = U \sqcap Z$.

Another important concept are splittings. They generalize several well-known constructions on sets.

Definition 4. Let $\mathcal{R}$ be a Dedekind category, and $Q : A \rightarrow A$ be partial equivalence relation, i.e. a symmetric idempotent relation such that $Q^\sim = Q$ and $Q; Q = Q$. An object B together with a relation $R : B \rightarrow A$ is called a splitting of Q (or R splits Q) iff $R; R^\sim = \mathbb{I}_B$ and $R^\sim; R = Q$. $\mathcal{R}$ has splittings iff all symmetric idempotent relations in $\mathcal{R}$ split.

A splitting is unique up to isomorphism. If Q is a partial identity the object B of the splitting corresponds to the subset given by Q . Analogously, if Q is an equivalence relation B corresponds to the set of equivalence classes.

The notion dual to a splitting of a difunctional relation is the notion of a tabulation [4].

Definition 5. Let $\mathcal{R}$ be a Dedekind category. A pair of maps $f : C \rightarrow A$ and $g : C \rightarrow B$ tabulates a relation $Q : A \rightarrow B$ iff $f^\sim; g = Q$ and $f; f^\sim \sqcap g; g^\sim = \mathbb{I}_C$. $\mathcal{R}$ is called tabular iff every relation is tabular, i.e. there is a pair a mappings tabulating the relation.

Notice, that tabulations are strongly related to the representability of the Dedekind category [\[4\]](#).

A tabulation of the greatest relation Π_{AB} is called a relational product of A and B . We use the notation $A \times B$ for the product object and $\pi : A \times B \rightarrow A$ and $\rho : A \times B \rightarrow B$ for the projections. A relational product constitutes a product in the subcategory of mappings, and is, therefore, an abstract counterpart of a cartesian product of sets. We use the notation $\langle Q, R \rangle := Q; \pi^\smile \sqcap R; \rho^\smile$ for relations $Q : C \rightarrow A$ and $R : C \rightarrow B$. Notice that if Q and R are mappings, this construction computes the unique product map induced by Q and R , i.e. the unique map $h : C \rightarrow A \times B$ with $h; \pi = Q$ and $h; \rho = R$. In addition, we use $S \times T := \langle \pi; S, \rho; T \rangle : A \times B \rightarrow C \times D$ for relations $S : A \times C$ and $T : B \rightarrow D$. Related to these two constructions we have in tabular Dedekind categories:

1. $\langle Q, R \rangle; \pi = Q$ if R is total, and $\langle Q, R \rangle; \rho = R$ if Q is total,
2. $\langle Q, R \rangle; (S \times T) = \langle Q; S, R; T \rangle$ for all $Q : C \rightarrow A$, $R : C \rightarrow B$, $S : A \rightarrow D$ and $T : B \rightarrow E$;
3. $(U \times V); (S \times T) = U; S \times V; T$ for all $U : C \rightarrow A$, $V : F \rightarrow B$, $S : A \rightarrow D$ and $T : B \rightarrow E$.

In the remainder of the paper we will use the properties above of relational products without mentioning.

Relational products are associative, i.e. the relation $\text{ass} : (A \times B) \times C \rightarrow A \times (B \times C)$ defined by $\text{ass} = \langle \pi; \pi, \langle \pi; \rho, \rho \rangle \rangle$ is a bijective function. Its converse (or inverse) is the relation $\text{ass}^\smile = \langle \langle \pi, \rho; \pi \rangle, \rho; \rho \rangle$. In addition, the family of the relations ass is a natural transformation between the relators $(\cdot \times \cdot) \times \cdot$ and $\cdot \times (\cdot \times \cdot)$, i.e. we have $\text{ass}; (Q \times (R \times S)) = ((Q \times R) \times S); \text{ass}$ for all suitable relations Q, R and S .

For further details on relational products we refer to any of [\[8,9,10\]](#).

An object 1 is called a unit iff $\mathbb{I}_1 = \Pi_{11}$ and Π_{A1} is total for all objects A . Notice that a unit is a terminal object in the subcategory of mappings. Therefore it is unique up to isomorphism and a neutral element for $\times$. In particular, the projection $\pi : A \times 1 \rightarrow A$ is an isomorphism.

In the remainder of this paper we will use the partial identity

$$i := \mathbb{I}_{(L_1 \times L_2) \times (L_2 \times L_3)} \sqcap \pi; \rho; \pi^\smile; \rho^\smile$$

and the univalent relation (partial function) $\text{com} : (L_1 \times L_2) \times (L_2 \times L_3) \rightarrow L_1 \times L_3$ defined by $\text{com} := i; (\pi \times \rho)$.

In the following lemma we have summarized some properties of the relations introduced so far.

Lemma 2. 1. $\text{ass}; (\mathbb{I}_{L_0 \times L_1} \times \text{com}); \text{com} = (\text{com} \times \mathbb{I}_{L_2 \times L_3}); \text{com}$,
 2. $\langle \mathbb{I}_{L_0 \times L_1}, \langle \rho, \rho \rangle \rangle; \text{com} = \mathbb{I}_{L_0 \times L_1}$,
 3. $\langle \langle \pi, \pi \rangle, \mathbb{I}_{L_0 \times L_1} \rangle; \text{com} = \mathbb{I}_{L_0 \times L_1}$.

Because of lack of space we omit the proof of this lemma.

3 Bisimulation

In this section we want to recall some basic properties of bisimulations in the relation algebraic framework [11]. The approach taken in that paper is quite general and covers a huge class of different bisimulations. In current we choose the behavior operation from [11] to be the identity.

Definition 6. Let $\mathcal{R}$ be a Dedekind category, $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator, and $P_1 : S_1 \rightarrow F(S_1)$ and $P_2 : S_2 \rightarrow F(S_2)$ be F coalgebras. A relation $\Phi : S_1 \rightarrow S_2$ is called a bisimulation (from P_1 and P_2) iff

$$\Phi^\smile ; P_1 \sqsubseteq P_2 ; F(\Phi^\smile) \text{ and } \Phi ; P_2 \sqsubseteq P_1 ; F(\Phi).$$

Using residuals the two inclusions can be rewritten as $\Phi \sqsubseteq \tau(\Phi)$ where

$$\tau(\Phi) := P_1^\smile \setminus (F(\Phi) ; P_2^\smile) \sqcap (P_1 ; F(\Phi)) / P_2.$$

The next lemma shows the basic properties of bisimulations in the relation algebraic context.

Lemma 3. Let $\mathcal{R}$ be a Dedekind category, $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator, $P_1 : S_1 \rightarrow F(S_1)$, $P_2 : S_2 \rightarrow F(S_2)$ and $P_3 : S_3 \rightarrow F(S_3)$ be F coalgebras, and $\Phi : S_1 \rightarrow S_2, \Psi : S_2 \rightarrow S_3$ and $\Phi_i : S_1 \rightarrow S_2$ ($i \in I$) be bisimulations. Then the following relations are bisimulations

1. $\perp_{AB}$,
2. $\mathbb{I}_A$,
3. $\Phi^\smile$,
4. $\Phi ; \Psi$,
5. $\bigsqcup_{i \in I} \Phi_i$.

The previous lemma shows that there is a greatest bisimulation (between two processes). Furthermore, τ is monotonic and, therefore, has a greatest fixed point Θ . Θ is also the greatest post fixed point, i.e. it satisfies $\Theta \sqsubseteq \tau(\Theta)$, and hence is the greatest bisimulation. Furthermore the existence of Θ is independent of the two processes. In order to call two processes bisimilar we have to require that Θ is total and surjective, i.e. that Θ relates the processes and all of their derivatives. In particular, we use the following definition.

Definition 7. Let $\mathcal{R}$ be a Dedekind category, $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator, and $P_1 : S_1 \rightarrow F(S_1)$ and $P_2 : S_2 \rightarrow F(S_2)$ be F coalgebras. Then

1. P_1 is called a future instance of P_2 , or P_1 is eventually bisimilar to P_2 , denoted by $P_1 \preceq P_2$, iff there is a total bisimulation from P_1 to P_2 ;
2. P_1 and P_2 are called bisimilar, denoted by $P_1 \approx P_2$, iff there is a total and surjective bisimulation between P_1 and P_2 .

Notice, that P_1 and P_2 are bisimilar iff the greatest fixed point Θ of τ is total and surjective. Furthermore, Θ is difunctional, i.e. $\Theta ; \Theta^\smile ; \Theta = \Theta$.

Lemma 4. Let $\mathcal{R}$ be a Dedekind category, and $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator. Then the relation $\preceq$ is a pre-ordering on the class of processes and its induced equivalence relation is $\approx$ i.e. $P_1 \preceq P_2$ and $P_2 \preceq P_1$ implies $P_1 \approx P_2$ for all F coalgebras $P_1 : S_1 \rightarrow F(S_1)$ and $P_2 : S_2 \rightarrow F(S_2)$.

Proof. The reflexivity and transitivity of $\preceq$ follows immediately from Lemma 3.2.&4. since the identity is total and the composition of total relations is total. Suppose $P_1 \preceq P_2$ and $P_2 \preceq P_1$, i.e. there is a total bisimulation $\Phi : S_1 \rightarrow S_2$ and a total bisimulation $\Psi : S_2 \rightarrow S_1$. By Lemma 3.3.&5. the relation $\Phi \sqcup \Psi^\sim$ is a bisimulation. This relation is total (since Φ is) and surjective (since Ψ is total). $\square$

Due to the last lemma the class of equivalence classes $[P]$ (with respect to $\approx$) of F coalgebras is ordered by $[P_1] \sqsubseteq [P_2]$ iff $P_1 \preceq P_2$. We denote by $(\mathcal{L}(F), \sqsubseteq)$ the ordered class of those equivalence classes. If the underlying Dedekind category has splittings, then each equivalence class has a canonical representative (see 1.1). Intuitively, the canonical representative is given by the maximal identified graph. Instead of using equivalence classes we may use the canonical representatives. The corresponding order structure is also denoted by $(\mathcal{L}(F), \sqsubseteq)$.

Lemma 5. *Let $\mathcal{R}$ be a Dedekind category with splittings, and $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator. Then each pair of F coalgebras $P_1 : S_1 \rightarrow F(S_1)$ and $P_2 : S_2 \rightarrow F(S_2)$ has a greatest lower bound with respect to $\preceq$.*

Proof. Let Θ be the greatest bisimulation from P_1 to P_2 . Then Θ is difunctional, and, hence, $\Theta; \Theta^\sim$ an equivalence relation. Suppose $R : S \rightarrow S_1$ splits Θ , i.e. we have $R; R^\sim = \mathbb{I}_S$ and $R^\sim; R = \Theta; \Theta^\sim$. Then R is total and the relation $P := R; P_1; F(R^\sim) : S \rightarrow F(S)$ is a F coalgebra. We compute

$$\begin{aligned}
 R; P_1 &= R; P_1; F(\mathbb{I}_{S_1}) \\
 &\sqsubseteq R; P_1; F(\Theta; \Theta^\sim) && \Theta; \Theta^\sim \text{ reflexive} \\
 &= R; P_1; F(R^\sim; R) && R \text{ splits } \Theta; \Theta^\sim \\
 &= R; P_1; F(R^\sim); F(R) \\
 &= P; F(R), \\
 R^\sim; P &= R^\sim; R; P_1; F(R^\sim) \\
 &= \Theta; \Theta^\sim; P_1; F(R^\sim) && R \text{ splits } \Theta; \Theta^\sim \\
 &\sqsubseteq F(\Theta); P_2; F(\Theta^\sim); F(R^\sim) && \Theta^\sim \text{ is a bisimulation} \\
 &\sqsubseteq P_1; F(\Theta); F(\Theta^\sim); F(R^\sim) && \Theta \text{ is a bisimulation} \\
 &= P_1; F(\Theta; \Theta^\sim; R^\sim) \\
 &= P_1; F(R^\sim; R; R^\sim) && R \text{ splits } \Theta; \Theta^\sim \\
 &= P_1; F(R^\sim). && R \text{ is a splitting}
 \end{aligned}$$

and conclude that R is a bisimulation from P to P_1 . The relation $R; \Theta$ is a bisimulation (from P to P_2) by Lemma 3. Furthermore, this relation is total since

$$\mathbb{I}_S = R; R^\sim; R; R^\sim = R; \Theta; \Theta^\sim; R^\sim = R; \Theta; (R; \Theta)^\sim.$$

Suppose $P' : S' \rightarrow F(S')$ is a F coalgebra with $P' \preceq P_1$ and $P' \preceq P_2$, i.e. there are total bisimulations $\Phi_1 : S' \rightarrow S_1$ and $\Phi_2 : S' \rightarrow S_2$. By Lemma 3 the relation

$\Phi_1^\sim; \Phi_2$ is bisimulation from P_1 to P_2 , and, thus, we have $\Phi_1^\sim; \Phi_2 \sqsubseteq \Theta$. This implies $\Phi_2 \sqsubseteq \Phi_1; \Phi_1^\sim; \Phi_2 \sqsubseteq \Phi_1; \Theta$ since Φ_1 is total. Furthermore, the relation $\Phi_1; R^\sim$ is a bisimulation from P' to P . It remains to show that this relation is total. We compute

$$\begin{aligned}
 \mathbb{I}_{S'} &\sqsubseteq \Phi_2; \Phi_2^\sim && \Phi_2 \text{ is total} \\
 &\sqsubseteq \Phi_1; \Theta; (\Phi_1; \Theta)^\sim && \text{see above} \\
 &= \Phi_1; \Theta; \Theta^\sim; \Phi_1^\sim \\
 &= \Phi_1; R^\sim; R; \Phi_1^\sim && R \text{ splits } \Theta; \Theta^\sim \\
 &= \Phi_1; R^\sim; (\Phi_1; R^\sim)^\sim.
 \end{aligned}$$

This completes the proof. $\square$

From the previous lemma we immediately obtain the following corollary.

Corollary 1. *Let $\mathcal{R}$ be a Dedekind category with splittings, and $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator. Then $(\mathcal{L}(F), \sqsubseteq)$ is a lower semi-lattice.*

We are interested in operation defined on the ordered classes $\mathcal{L}(F)$ of F coalgebras. As an intermediate step the following category turns out to be useful.

Definition 8. *Let $\mathcal{R}$ be a Dedekind category, and $F : \mathcal{R} \rightarrow \mathcal{R}$ be an endorelator. The category $\text{BSIM}(F)$ has F coalgebras as objects and bisimulations as morphisms.*

From Lemma 3 we conclude that $\text{BSIM}(F)$ is an ordered category with converse. Notice that a relator $F : \text{BSIM}(G) \rightarrow \text{BSIM}(H)$ respects the pre-order $\preceq$ and the equivalence relation $\approx$. Therefore, it induces a monotone function $F^L : \mathcal{L}(G) \rightarrow \mathcal{L}(H)$. A generalization to multiple parameters is obvious.

The next lemma provides a method to lift a relator between the underlying Dedekind categories to a relator between coalgebras.

Lemma 6. *Let $\mathcal{R}_1, \mathcal{R}_2$ be a Dedekind categories, $G : \mathcal{R}_1 \rightarrow \mathcal{R}_1$ and $H : \mathcal{R}_2 \rightarrow \mathcal{R}_2$ be endorelators, $F : \mathcal{R}_1 \rightarrow \mathcal{R}_2$ be a relator, and $\eta : F \circ G \rightarrow H \circ F$ a lax natural transformation. Then F_η defined by*

- $F_\eta(P) := F(P); \eta,$
- $F_\eta(\Phi) := F(\Phi),$

is a relator from $\text{BSIM}(G)$ to $\text{BSIM}(H)$.

Proof. First of all, if $P : S \rightarrow G(S)$, then we have $F(P); \eta : F(S) \rightarrow H(F(S))$, i.e. $F_\eta(P)$ is a H coalgebra. It remains to verify that if Φ is a bisimulation between G coalgebras P_1 and P_2 , then $F(\Phi)$ is a bisimulation between $F(P_1); \eta$ and

$F(P_2); \eta$. All other properties follow from the fact the operations on morphisms in $\mathcal{R}_2$ and $\text{BSIM}(H)$ are the same. Consider the following computation:

$$\begin{aligned}
 F(\Phi); F(P_2); \eta &= F(\Phi; P_2); \eta \\
 &\sqsubseteq F(P_1; G(\Phi)); \eta && \Phi \text{ bisimulation and } F \text{ relator} \\
 &= F(P_1); F(G(\Phi)); \eta \\
 &\sqsubseteq F(P_1); \eta; H(F(\Phi)). && \eta \text{ lax nat. trans.}
 \end{aligned}$$

The other inclusion follows analogously. $\square$

Again, a generalization of the previous lemma to multiple parameters is straightforward.

If we combine the previous lemma with the observation above, we get the following corollary.

Corollary 2. *Let $\mathcal{R}_1, \mathcal{R}_2$ be a Dedekind categories, $G : \mathcal{R}_1 \rightarrow \mathcal{R}_1$ and $H : \mathcal{R}_2 \rightarrow \mathcal{R}_2$ be endorelators, $F : \mathcal{R}_1 \rightarrow \mathcal{R}_2$ be a relator, and $\eta : F \circ G \rightarrow H \circ F$ a lax natural transformation. Then $F_\eta^L : \mathcal{L}(G) \rightarrow \mathcal{L}(H)$ is a monotone function.*

4 The Category $\text{SProc}(F)$

In order to define a category of processes as morphisms we fix a binary relator $F : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$. The first parameter of the relator is considered to be the object of actions (or labels) performed by a process. Fixing an object L we obtain an endorelator $F(L, \cdot)$. We call the elements of $\mathcal{L}(F(L, \cdot))$ processes of kind L . Recall that processes are either equivalence classes of coalgebras or, alternatively, the canonical representative of such an equivalence class. An object in $\text{SProc}(F)$ is an object L from $\mathcal{R}$. A morphism in $\text{SProc}(F)$ between two objects L_1 and L_2 is a process of kind $L_1 \times L_2$. Recall that every class of morphisms in $\text{SProc}(F)$ is lower semi-lattice.

The standard example of the a structure $\text{SProc}(F)$ is given by an arbitrary Dedekind category with splitting and relational products and the binary relator $F(L, S) := L \times S$.

Composition in $\text{SProc}(F)$ is (synchronous) parallel composition $+$ hiding of internal channels. Therefore, we study both concepts separately and start with parallel composition.

Suppose $\alpha_{L_1, L_2, S_1, S_2}$ is a family a relations from $F(L_1, S_1) \times F(L_2, S_2) \rightarrow F(L_1 \times L_2, S_1 \times S_2)$. In the following we will omit the index of α and define $P_1 \parallel P_2 := (P_1 \times P_2); \alpha$. Throughout the paper we will require several properties of α . If we denote by $\mathbf{1}_L : 1 \rightarrow F(L \times L, 1)$ the relation

$$\mathbf{1}_L := \prod_{1F(L \times L, 1)}; F(\mathbb{I}_{L \times L} \sqcap \pi; \rho^\sim, \mathbb{I}_1),$$

then we state the properties on α as follows:

($\alpha 1$) is a lax natural transformation between the relators $F(L_1, \cdot) \times F(L_2, \cdot)$ and $F(L_1 \times L_2, \cdot \times \cdot)$, i.e. $(F(\mathbb{I}_{L_1}, Q) \times F(\mathbb{I}_{L_2}, R)); \alpha \sqsubseteq \alpha; F(\mathbb{I}_{L_1 \times L_2}, Q \times R)$,

- $(\alpha 2)$ is a natural transformation between $F(\cdot, S_1) \times F(\cdot, S_2)$ and $F(\cdot \times \cdot, S_1 \times S_2)$,
 i.e. $(F(U, \mathbb{I}_{S_1}) \times F(V, \mathbb{I}_{S_2})); \alpha = \alpha; F(U \times V, \mathbb{I}_{S_1 \times S_2})$,
 $(\alpha 3)$ $\text{ass}; (\mathbb{I} \times \alpha); \alpha = (\alpha \times \mathbb{I}); \alpha; F(\text{ass}, \text{ass})$,
 $(\alpha 4a)$ $(\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \alpha; F(i, \pi) = \pi; F(\langle \mathbb{I}_{L_1 \times L_2}, \langle \rho, \rho \rangle \rangle; i, \mathbb{I}_S)$ for all objects L_1, L_2 and S ,
 $(\alpha 4b)$ $(\mathbf{1}_{L_1} \times \mathbb{I}_{(L_1 \times L_2) \times S}); \alpha; F(i, \rho) = \rho; F(\langle \langle \pi, \pi \rangle, \mathbb{I}_{L_1 \times L_2} \rangle; i, \mathbb{I}_S)$ for all objects L_1, L_2 and S .

$(\alpha 1)$ ensures that parallel composition is a monotone operation between the lower semi-lattices $\mathcal{L}(F(L_1, \cdot)) \times \mathcal{L}(F(L_2, \cdot))$ and $\mathcal{L}(F(L_1 \times L_2, \cdot))$ since it is defined as the lifting of the product relator using α .

In terms of the actions (or the labels) we will require the stronger property $(\alpha 2)$, i.e. that α is a natural transformation. This property as well as $(\alpha 3)$ will ensure that composition is associative.

The last property $(\alpha 4)$ is used to prove that the process $\mathbf{1}_L$ appearing on the left side of the equations is a left and right identity for composition.

In the standard example α is given by $\langle \pi \times \pi, \rho \times \rho \rangle : (L_1 \times S_1) \times (L_2 \times S_2) \rightarrow (L_1 \times L_2) \times (S_1 \times S_2)$. The next lemma shows that this α satisfies all of the properties above.

Lemma 7. *Let $\mathcal{R}$ be a Dedekind category with relational products, and denote by $F(L, S) := L \times S$ the product relator. Then the family of relations $\alpha := \langle \pi \times \pi, \rho \times \rho \rangle$ satisfies the properties $(\alpha 1) - (\alpha 4)$.*

Proof. First of all, we have the following property $(*)$

$$\begin{aligned}
 & \langle (\pi; U^\sim) \times (\pi; Q^\sim), (\rho; V^\sim) \times (\rho; R^\sim) \rangle \\
 &= (\pi; \pi; U^\sim; \pi^\sim \sqcap \rho; \pi; Q^\sim; \rho^\sim); \pi^\sim \sqcap (\pi; \rho; V^\sim; \pi^\sim \sqcap \rho; \rho; R^\sim; \rho^\sim); \rho^\sim \\
 &= \pi; \pi; U^\sim; \pi^\sim; \pi^\sim \sqcap \rho; \pi; Q^\sim; \rho^\sim; \pi^\sim \sqcap \pi; \rho; V^\sim; \pi^\sim; \rho^\sim \sqcap \rho; \rho; R^\sim; \rho^\sim; \rho^\sim \\
 & \quad \text{by Lemma 1(2)} \\
 &= \pi; (\pi; U^\sim; \pi^\sim; \pi^\sim \sqcap \rho; V^\sim; \pi^\sim; \rho^\sim) \sqcap \rho; (\pi; Q^\sim; \rho^\sim; \pi^\sim \sqcap \rho; R^\sim; \rho^\sim; \rho^\sim) \\
 & \quad \text{by Lemma 1(2)} \\
 &= \pi; ((\pi; U) \times (\pi; V))^\sim \sqcap \rho; ((\rho; Q) \times (\rho; R))^\sim \\
 &= \langle (\pi; U) \times (\pi; V), (\rho; Q) \times (\rho; R) \rangle^\sim
 \end{aligned}$$

for all suitable Q, R, U and V .

$(\alpha 1)$ and $(\alpha 2)$ We compute

$$\begin{aligned}
 & (F(U, Q) \times F(V, R)); \alpha \\
 &= ((U \times Q) \times (V \times R)); \langle \pi \times \pi, \rho \times \rho \rangle \\
 &= ((U \times Q) \times (V \times R)); \langle \pi \times \pi, \rho \times \rho \rangle^\sim \quad \text{by } (*) \\
 &= ((\pi \times \pi, \rho \times \rho); ((U^\sim \times Q^\sim) \times (V^\sim \times R^\sim)))^\sim
 \end{aligned}$$

$$\begin{aligned}
&= \langle (\pi \times \pi); (U^\sim \times Q^\sim), (\rho \times \rho); (V^\sim \times R^\sim) \rangle^\sim \\
&= \langle (\pi; U^\sim) \times (\pi; Q^\sim), (\rho; V^\sim) \times (\rho; R^\sim) \rangle^\sim \\
&= \langle (\pi; U) \times (\pi; V), (\rho; Q) \times (\rho; R) \rangle && \text{by } (*) \\
&= \langle (\pi \times \pi); (U \times V), (\rho \times \rho); (Q \times R) \rangle \\
&= \langle \pi \times \pi, \rho \times \rho \rangle; ((U \times V) \times (Q \times R)) \\
&= \alpha; F(U \times V, Q \times R)
\end{aligned}$$

verifying $(\alpha 1)$ and $(\alpha 2)$.

$(\alpha 3)$ Consider the following computation

$$\begin{aligned}
(\mathbb{I} \times \alpha); \alpha &= (\mathbb{I} \times \alpha); \langle \pi \times \pi, \rho \times \rho \rangle^\sim && \text{by } (*) \\
&= (\langle \pi \times \pi, \rho \times \rho \rangle; (\mathbb{I} \times \alpha^\sim))^\sim \\
&= \langle \pi \times \pi, (\rho \times \rho); \alpha^\sim \rangle^\sim \\
&= ((\pi; \pi; \pi^\sim \sqcap \rho; \pi; \rho^\sim); \pi^\sim \sqcap (\pi; \rho; \pi^\sim \sqcap \rho; \rho^\sim); \alpha; \rho^\sim)^\sim \\
&= (\pi; \pi; \pi^\sim; \pi^\sim \sqcap \rho; \pi; \rho^\sim; \pi^\sim \sqcap \pi; \rho; \pi^\sim; \alpha; \rho^\sim \sqcap \rho; \rho^\sim; \alpha; \rho^\sim)^\sim \\
&\quad \text{by Lemma } \textcolor{red}{\Pi}(3) \\
&= (\pi; (\pi; \pi^\sim; \pi^\sim \sqcap \rho; \pi^\sim; \alpha; \rho^\sim) \sqcap \rho; (\pi; \rho^\sim; \pi^\sim \sqcap \rho; \rho^\sim; \alpha; \rho^\sim))^\sim \\
&\quad \text{by Lemma } \textcolor{red}{\Pi}(3) \\
&= (\pi; (\pi^\sim \times (\pi^\sim; \alpha)) \sqcap \rho; (\rho^\sim \times (\rho^\sim; \alpha))^\sim) \\
&= \langle \pi \times (\alpha^\sim; \pi), \rho \times (\alpha^\sim; \rho) \rangle \\
&= \langle \pi \times (\langle \pi \times \pi, \rho \times \rho \rangle; \pi), \rho \times (\langle \pi \times \pi, \rho \times \rho \rangle; \rho) \rangle && \text{by } (*) \\
&= \langle \pi \times (\pi \times \pi), \rho \times (\rho \times \rho) \rangle.
\end{aligned}$$

Analogously, we obtain $(\alpha \times \mathbb{I}); \alpha = \langle ((\pi \times \pi) \times \pi), ((\rho \times \rho) \times \rho) \rangle$, and conclude

$$\begin{aligned}
&\text{ass}; (\mathbb{I} \times \alpha); \alpha \\
&= \text{ass}; \langle \pi \times (\pi \times \pi), \rho \times (\rho \times \rho) \rangle && \text{see above} \\
&= \langle \text{ass}; (\pi \times (\pi \times \pi)), \text{ass}; (\rho \times (\rho \times \rho)) \rangle && \text{Lemma } \textcolor{red}{\Pi}(2) \\
&= \langle ((\pi \times \pi) \times \pi); \text{ass}, ((\rho \times \rho) \times \rho); \text{ass} \rangle && \text{ass nat. trans.} \\
&= \langle ((\pi \times \pi) \times \pi), ((\rho \times \rho) \times \rho); (\text{ass} \times \text{ass}) \rangle \\
&= (\alpha \times \mathbb{I}); \alpha; (\text{ass} \times \text{ass}) && \text{see above} \\
&= (\alpha \times \mathbb{I}); \alpha; F(\text{ass}, \text{ass}).
\end{aligned}$$

$(\alpha 4)$ First we obtain the following equation

$$\begin{aligned}
\langle (\pi \times \pi); i, \pi; \rho \rangle &= (\pi \times \pi); i; \pi^\sim \sqcap \pi; \rho^\sim \\
&= (\pi; \pi; \pi^\sim \sqcap \rho; \pi; \rho^\sim); i; \pi^\sim \sqcap \pi; \rho^\sim
\end{aligned}$$

$$\begin{aligned}
&= \pi; \pi; \pi^\sim; i; \pi^\sim \sqcap \rho; \pi; \rho^\sim; i; \pi^\sim \sqcap \pi; \rho; \rho^\sim && \text{Lemma III(2)} \\
&= \pi; (\pi; \pi^\sim; i; \pi^\sim \sqcap \rho; \rho^\sim) \sqcap \rho; \pi; \rho^\sim; i; \pi^\sim && \text{Lemma III(2)} \\
&= \pi; ((\pi^\sim; i) \times \mathbb{I}) \sqcap \rho; \pi; \rho^\sim; i; \pi^\sim \\
&= \langle (i; \pi) \times \mathbb{I}, \pi; i; \rho; \pi^\sim \rangle^\sim.
\end{aligned}$$

Then we conclude

$$\begin{aligned}
&(\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \alpha; F(i, \pi) \\
&= (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \langle \pi \times \pi, \rho \times \rho \rangle; (i \times \pi) \\
&= (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \langle (\pi \times \pi); i, (\rho \times \rho); \pi \rangle \\
&= (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \langle (\pi \times \pi); i, \pi; \rho \rangle \\
&= (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \langle (i; \pi) \times \mathbb{I}_S, \pi; i; \rho; \pi^\sim \rangle^\sim && \text{see above} \\
&= (\langle (i; \pi) \times \mathbb{I}_S, \pi; i; \rho; \pi^\sim \rangle; (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}))^\sim \\
&= \langle (i; \pi) \times \mathbb{I}_S, \pi; i; \rho; \pi^\sim; \mathbf{1}_{L_2} \rangle^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; \rho; \pi^\sim; \mathbf{1}_{L_2}; \rho^\sim)^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; \rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim \sqcap \pi; i; \rho; \pi^\sim; \mathbf{1}_{L_2}; \rho^\sim)^\sim
\end{aligned}$$

where the last = follows from

$$\begin{aligned}
((i; \pi) \times \mathbb{I}_S); \pi^\sim &= (\pi; i; \pi; \pi^\sim \sqcap \rho; \rho^\sim); \pi^\sim \\
&\sqsubseteq \pi; i; \pi; \pi^\sim; \pi^\sim \\
&= \pi; i^\sim; \pi; \pi^\sim; \pi^\sim && \text{partial identity} \\
&\sqsubseteq \pi; \rho; \pi; \rho^\sim; \pi^\sim; \pi; \pi^\sim; \pi^\sim \\
&\sqsubseteq \pi; \rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim && \pi \text{ univalent}
\end{aligned}$$

Together with

$$\begin{aligned}
&\pi^\sim; \mathbf{1}_{L_2}^\sim \\
&= \pi^\sim; ((\mathbb{I}_{L_2 \times L_2} \sqcap \pi; \rho^\sim) \times \mathbb{I}_1); \mathbb{I}_{(L_2 \times L_2) \times 1, 1} \\
&= \pi^\sim; (\pi; (\mathbb{I}_{L_2 \times L_2} \sqcap \pi; \rho^\sim); \pi^\sim \sqcap \rho; \rho^\sim); \mathbb{I}_{(L_2 \times L_2) \times 1, 1} \\
&= \pi^\sim; (\pi; \pi^\sim \sqcap \pi; \pi; \rho^\sim; \pi^\sim \sqcap \rho; \rho^\sim); \mathbb{I}_{(L_2 \times L_2) \times 1, 1} && \text{Lemma III(2)} \\
&= \pi^\sim; (\mathbb{I}_{(L_2 \times L_2) \times 1} \sqcap \pi; \pi; \rho^\sim; \pi^\sim); \mathbb{I}_{(L_2 \times L_2) \times 1, 1} \\
&= (\pi^\sim \sqcap \pi; \rho^\sim; \pi^\sim); \mathbb{I}_{(L_2 \times L_2) \times 1, 1} && \text{Lemma III(3)} \\
&= (\mathbb{I}_{L_2 \times L_2} \sqcap \pi; \rho^\sim); \pi^\sim; \mathbb{I}_{(L_2 \times L_2) \times 1, 1} && \text{Lemma III(2)} \\
&= (\mathbb{I}_{L_2 \times L_2} \sqcap \pi; \rho^\sim); \mathbb{I}_{L_2 \times L_2, 1} \\
&= (\rho \sqcap \pi); \mathbb{I}_{L_2 \times L_2, 1} && \text{Lemma III(1)}
\end{aligned}$$

we obtain

$$\begin{aligned}
& (\mathbb{I}_{(L_1 \times L_2) \times S} \times \mathbf{1}_{L_2}); \alpha; F(i, \pi) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; \rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim \sqcap \pi; i; \rho; \pi^\sim; \mathbf{1}_{L_2}^\sim; \rho^\sim)^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; \rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim \sqcap \pi; i; \rho; (\rho \sqcap \pi); \mathbb{I})^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; (\rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim \sqcap i; \rho; (\rho \sqcap \pi); \mathbb{I}))^\sim && \text{Lemma } \textcolor{red}{\text{II}}(2) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi; \rho^\sim; \pi^\sim; \pi^\sim \sqcap i; \rho; (\rho \sqcap \pi); \mathbb{I}))^\sim && \text{Lemma } \textcolor{red}{\text{II}}(4) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi \sqcap i; \rho; (\rho \sqcap \pi); \mathbb{I}); \rho^\sim; \pi^\sim; \pi^\sim)^\sim && \text{Lemma } \textcolor{red}{\text{II}}(1) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi \sqcap i; (\rho; \rho \sqcap \rho; \pi); \mathbb{I}); \rho^\sim; \pi^\sim; \pi^\sim)^\sim && \text{Lemma } \textcolor{red}{\text{II}}(2) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi \sqcap (\rho; \rho \sqcap i; \rho; \pi); \mathbb{I}); \rho^\sim; \pi^\sim; \pi^\sim)^\sim && \text{Lemma } \textcolor{red}{\text{II}}(4) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi \sqcap (\rho; \rho \sqcap i; \rho; \pi); \mathbb{I}); \rho^\sim; \pi^\sim; \pi^\sim)^\sim
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
& \pi; F(\langle \mathbb{I}_{L_1 \times L_2}, \langle \rho, \rho \rangle \rangle; i; \mathbb{I}_S) \\
&= \pi; (\langle \langle \mathbb{I}_{L_1 \times L_2}, \langle \rho, \rho \rangle \rangle; i) \times \mathbb{I}_S) \\
&= \pi; (((\pi^\sim \sqcap \langle \rho, \rho \rangle; \rho^\sim); i) \times \mathbb{I}_S) \\
&= \pi; (\pi; (\pi^\sim \sqcap \langle \rho, \rho \rangle; \rho^\sim); i; \pi^\sim \sqcap \rho; \rho^\sim) \\
&= \pi; (\pi; \pi^\sim; i; \pi^\sim \sqcap \pi; \langle \rho, \rho \rangle; \rho^\sim; i; \pi^\sim \sqcap \rho; \rho^\sim) && \text{Lemma } \textcolor{red}{\text{II}}(2) \\
&= \pi; (((\pi^\sim; i) \times \mathbb{I}_S) \sqcap \pi; \langle \rho, \rho \rangle; \rho^\sim; i; \pi^\sim) \\
&= \pi; ((\pi^\sim; i) \times \mathbb{I}_S) \sqcap \pi; \pi; \langle \rho, \rho \rangle; \rho^\sim; i; \pi^\sim && \text{Lemma } \textcolor{red}{\text{II}}(2) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; \rho; \langle \rho, \rho \rangle^\sim; \pi^\sim; \pi^\sim)^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; \rho; (\pi; \rho^\sim \sqcap \rho; \rho^\sim); \pi^\sim; \pi^\sim)^\sim \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (\rho; \pi \sqcap \rho; \rho); \rho^\sim; \pi^\sim; \pi^\sim)^\sim && \text{Lemma } \textcolor{red}{\text{II}}(2) \\
&= (((i; \pi) \times \mathbb{I}_S); \pi^\sim \sqcap \pi; i; (i; \rho; \pi \sqcap \rho; \rho); \rho^\sim; \pi^\sim; \pi^\sim)^\sim. && \text{Lemma } \textcolor{red}{\text{II}}(4)
\end{aligned}$$

It remains to show that $\rho; \pi \sqcap (\rho; \rho \sqcap i; \rho; \pi); \mathbb{I} = i; \rho; \pi \sqcap \rho; \rho$. First, we have

$$\begin{aligned}
i; \rho; \pi &= (\mathbb{I}_{(L_1 \times L_2) \times (L_2 \times L_3)} \sqcap \pi; \rho; \pi^\sim; \rho^\sim); \rho; \pi \\
&= (\rho \sqcap \pi; \rho; \pi^\sim); \pi && \text{Lemma } \textcolor{red}{\text{II}}(3) \\
&= \rho; \pi \sqcap \pi; \rho && \text{Lemma } \textcolor{red}{\text{II}}(3)
\end{aligned}$$

so that

$$\begin{aligned}
\rho; \pi \sqcap (\rho; \rho \sqcap i; \rho; \pi); \mathbb{I} &= \rho; \pi \sqcap (\rho; \rho \sqcap \rho; \pi \sqcap \pi; \rho); \mathbb{I} \\
&= \rho; \rho \sqcap \rho; \pi \sqcap \pi; \rho && \text{Lemma } \textcolor{red}{\text{II}}(5) \\
&= \rho; \rho \sqcap i; \rho; \pi
\end{aligned}$$

follows. This completes the proof. $\square$

Notice that the specific α of the previous lemma satisfies even stronger properties. For example, α is a natural transformation in all four parameters and each individual α is an isomorphism.

We model hiding (as well as relabelling) by a partial function $h : L_1 \rightarrow L_2$ on labels, and we define $P \setminus h := P; F(h, \mathbb{I}_S)$ for $P : S \rightarrow F(L, S)$. Notice that this operation is defined by lifting the identity relator using $F(h, \mathbb{I})$. It remains to show that this is a lax natural transformation from $F(L_1, \cdot)$ to $F(L_2, \cdot)$, which follows immediately from

$$F(\mathbb{I}_{L_1}, Q); F(h, \mathbb{I}_{S_1}) = F(h, Q) = F(h, \mathbb{I}_{S_2}); F(\mathbb{I}_{L_2}, Q).$$

Suppose $P_1 : S_1 \rightarrow F(L_1 \times L_2, S)$ and $P_2 : S_2 \rightarrow F(L_2 \times L_3, S_2)$ are representatives of processes in $\text{SProc}(F)$. Its parallel composition $P_1 \parallel P_2 : S_1 \times S_2 \rightarrow F((L_1 \times L_2) \times (L_2 \times L_3), S_1 \times S_2)$ still omits the internal channels from L_2 . The partial function $\text{com} : (L_1 \times L_2) \times (L_2 \times L_3) \rightarrow L_1 \times L_3$ hides those channels and allows an action in the composition if the internal channels match. We define $P_1 \star P_2 := (P_1 \parallel P_2) \setminus \text{com}$. Since composition is defined in terms of parallel composition and hiding it is a monotone operation between the corresponding lower semi-lattices.

Lemma 8. *The processes $P_1 \star (P_2 \star P_3)$ and $(P_1 \star P_2) \star P_3$ are bisimilar.*

Proof. First of all, we have

$$\begin{aligned} & P_1 \star (P_2 \star P_3) \\ &= (P_1 \times ((P_2 \times P_3); \alpha; F(\text{com}, \mathbb{I}))); \alpha; F(\text{com}, \mathbb{I}) \\ &= (P_1 \times (P_2 \times P_3)); (\mathbb{I} \times \alpha); (\mathbb{I} \times F(\text{com}, \mathbb{I})); \alpha; F(\text{com}; \mathbb{I}) \\ &= (P_1 \times (P_2 \times P_3)); (\mathbb{I} \times \alpha); (F(\mathbb{I}, \mathbb{I}) \times F(\text{com}, \mathbb{I})); \alpha; F(\text{com}; \mathbb{I}) \\ &= (P_1 \times (P_2 \times P_3)); (\mathbb{I} \times \alpha); \alpha; F(\mathbb{I} \times \text{com}, \mathbb{I}); F(\text{com}; \mathbb{I}) \quad \text{by } (\alpha 2) \\ &= (P_1 \times (P_2 \times P_3)); (\mathbb{I} \times \alpha); \alpha; F((\mathbb{I} \times \text{com}); \text{com}, \mathbb{I}) \end{aligned}$$

and analogously $(P_1 \star P_2) \star P_3 = ((P_1 \times P_2) \times P_3); (\alpha \times \mathbb{I}); \alpha; F((\text{com} \times \mathbb{I}); \text{com}, \mathbb{I})$. This implies

$$\begin{aligned} & \text{ass}; (P_1 \star (P_2 \star P_3)) \\ &= \text{ass}; (P_1 \times (P_2 \times P_3)); (\mathbb{I} \times \alpha); \alpha; F((\mathbb{I} \times \text{com}); \text{com}, \mathbb{I}) \\ &= ((P_1 \times P_2) \times P_3); \text{ass}; (\mathbb{I} \times \alpha); \alpha; F((\mathbb{I} \times \text{com}); \text{com}, \mathbb{I}) \quad \text{ass nat. trans.} \\ &= ((P_1 \times P_2) \times P_3); (\alpha \times \mathbb{I}); \alpha; F(\text{ass}, \text{ass}); F((\mathbb{I} \times \text{com}); \text{com}, \mathbb{I}) \quad \text{by } (\alpha 3) \\ &= ((P_1 \times P_2) \times P_3); (\alpha \times \mathbb{I}); \alpha; F(\text{ass}; (\mathbb{I} \times \text{com}); \text{com}, \text{ass}) \\ &= ((P_1 \times P_2) \times P_3); (\alpha \times \mathbb{I}); \alpha; F((\text{com} \times \mathbb{I}); \text{com}, \text{ass}) \quad \text{Lemma 2(1)} \\ &= ((P_1 \times P_2) \times P_3); (\alpha \times \mathbb{I}); \alpha; F((\text{com} \times \mathbb{I}); \text{com}, \mathbb{I}); F(\mathbb{I}, \text{ass}) \\ &= ((P_1 \star P_2) \star P_3); F(\mathbb{I}, \text{ass}). \end{aligned}$$

From the fact that ass is an isomorphism we get

$$\begin{aligned}
& \text{ass}^\sim; ((P_1 \bullet P_2) \bullet P_3) \\
&= \text{ass}^\sim; ((P_1 \bullet P_2) \bullet P_3); F(\mathbb{I}, \text{ass}; \text{ass}^\sim) && \text{ass isomorphism} \\
&= \text{ass}^\sim; ((P_1 \bullet P_2) \bullet P_3); F(\mathbb{I}, \text{ass}); F(\mathbb{I}, \text{ass}^\sim) \\
&= \text{ass}^\sim; \text{ass}; (P_1 \bullet (P_2 \bullet P_3)); F(\mathbb{I}, \text{ass}^\sim) && \text{see above} \\
&= (P_1 \bullet (P_2 \bullet P_3)); F(\mathbb{I}, \text{ass}^\sim) && \text{ass isomorphism}
\end{aligned}$$

verifying that ass is a bisimulation between the processes $(P_1 \bullet P_2) \bullet P_3$ and $P_1 \bullet (P_2 \bullet P_3)$, which is total and surjective, of course. $\square$

As already mentioned above the process $\mathbf{1} : 1 \rightarrow F(L \times L, 1)$ defined by $\mathbf{1} := \mathbb{I}; F(\mathbb{I} \sqcap \pi; \rho^\sim, \mathbb{I})$ is the identity element for composition.

Lemma 9. $P \bullet \mathbf{1}$ and $\mathbf{1} \bullet P$ are bisimilar to P .

Proof. We are going to show that $\pi : S \times 1 \rightarrow S$ is a bisimulation between $P \bullet \mathbf{1}$ and P . Since $\mathbf{1}$ is a unit π is an isomorphism, and, hence, total and surjective. The first inclusion follows from

$$\begin{aligned}
\pi; P &= \pi; P; F(\langle \mathbb{I}, \langle \rho, \rho \rangle \rangle; \text{com}, \mathbb{I}) && \text{Lemma 2(2)} \\
&= \pi; P; F(\langle \mathbb{I}, \langle \rho, \rho \rangle \rangle; i; \text{com}, \mathbb{I}) && \text{Lemma 4(4)} \\
&= \pi; P; F(\langle \mathbb{I}, \langle \rho, \rho \rangle \rangle; i, \mathbb{I}); F(\text{com}, \mathbb{I}) \\
&= (P \times \mathbb{I}); \pi; F(\langle \mathbb{I}, \langle \rho, \rho \rangle \rangle; i, \mathbb{I}); F(\text{com}, \mathbb{I}) \\
&= (P \times \mathbb{I}); (\mathbb{I} \times \mathbf{1}); \alpha; F(i, \pi); F(\text{com}, \mathbb{I}) && \text{by } (\alpha 4a) \\
&= (P \times \mathbf{1}); \alpha; F(i; \text{com}, \mathbb{I}); F(\mathbb{I}, \pi) \\
&= (P \times \mathbf{1}); \alpha; F(\text{com}, \mathbb{I}); F(\mathbb{I}, \pi) && \text{Lemma 4(4)} \\
&= (P \bullet \mathbf{1}); F(\mathbb{I}, \pi).
\end{aligned}$$

For the second inclusion consider

$$\begin{aligned}
\pi^\sim; (P \bullet \mathbf{1}) &= \pi^\sim; (P \bullet \mathbf{1}); F(\mathbb{I}, \pi; \pi^\sim) && \pi \text{ isomorphism} \\
&= \pi^\sim; (P \bullet \mathbf{1}); F(\mathbb{I}, \pi); F(\mathbb{I}, \pi^\sim) \\
&= \pi^\sim; \pi; P; F(\mathbb{I}, \pi^\sim) && \text{see above} \\
&= P; F(\mathbb{I}, \pi^\sim). && \pi \text{ isomorphism}
\end{aligned}$$

The fact that $\mathbf{1}$ is also a left identity element is shown analogously using $(\alpha 4b)$ and Lemma 2(3). $\square$

From previous two lemmas we get the main result of this paper as a corollary.

Corollary 3. $\text{SProc}(F)$ is an ordered category.

5 Future Work

As already mentioned in the introduction we see this paper as a starting point of a detailed investigation of the structure of $\text{SProc}(F)$. Additional relational operations such as converse and union will be defined, and we are going to compare their properties with the axioms of an allegory. In addition we want to study time related operations such as the unit delay operation.

Corresponding to guarded processes we are going to study suitable notions of guarded relators on $\text{SProc}(F)$. The existence and the uniqueness of fixed points of such a relator is of special interest for recursively defined processes.

Last but not least, we want to use $\text{SProc}(F)$ to define a denotational semantics for a synchronous version of CCS.

References

1. Abramsky, S., Gay, S., Nagarajan, R.: Interaction Categories and the Foundations of Typed Concurrent Programming. In: Broy, M. (ed.) Proceedings of the 1994 Marktoberdorf Summer School on Deductive Program Design, pp. 35–113. Springer, Heidelberg (1996)
2. Aczel, P.: Non-Well-Founded Sets. CSLI Publication, Stanford, CA (1988)
3. Bird, R., de Moor, O.: Algebra of Programming. Prentice-Hall, Englewood Cliffs (1997)
4. Freyd, P., Scedrov, A.: Categories, Allegories. North-Holland, Amsterdam (1990)
5. Kawahara, Y.: Notes on the universality of relational functors, Memoirs of the Faculty of Science. Kyushu University, vol. 27(3), pp. 275–289 (1973)
6. Olivier, J.P., Serrato, D.: Catégories de Dedekind. Morphismes dans les Catégories de Schröder. C.R. Acad. Sci. Paris 290, 939–941 (1980)
7. Olivier, J.P., Serrato, D.: Squares and Rectangles in Relational Categories - Three Cases: Semilattice, Distributive lattice and Boolean Non-unitary. Fuzzy sets and systems 72, 167–178 (1995)
8. Schmidt, G., Ströhlein, T.: Relationen und Graphen. Springer, Heidelberg (1989); English version: Relations and Graphs. Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer (1993)
9. Schmidt, G., Hattensperger, C., Winter, M.: Heterogeneous Relation Algebras. In: Brink, C., Kahl, W., Schmidt, G. (eds.) Relational Methods in Computer Science. Advances in Computer Science, Springer, Heidelberg (1997)
10. Winter, M.: A relation algebraic Approach to Interaction Categories. Information Sciences 119, 301–314 (1999)
11. Winter, M.: A Relation-Algebraic Theory of Bisimulations (submitted to Fundamenta Informatica)

Automatic Proof Generation in Kleene Algebra

James Worthington

Mathematics Department, Cornell University
Ithaca, NY 14853-4201 USA
worthing@math.cornell.edu

Abstract. In this paper, we develop the basic theory of *disimulations*, a type of relation between two automata which witnesses equivalence. We show that many standard constructions in the theory of automata such as determinization, minimization, inaccessible state removal, et al., are instances of disimilar automata. Then, using disimulations, we define an “algebraic” proof system for the equational theory of Kleene algebra in which a proof essentially consists of a sequence of matrices encoding automata and disimulations between them. We show that this proof system is complete for the equational theory of Kleene algebra, and that proofs in this system can be constructed by a *PSPACE* transducer.

1 Introduction

The class of Kleene algebras (KA) is defined by equations and equational implications over the signature $\{0, 1, +, \cdot, *\}$. Well-known Kleene algebras include relational algebras, trace algebras, and sets of regular languages. In fact, the set of regular languages over an alphabet Σ is the free Kleene algebra on Σ [3]. A Kleene algebra with tests (KAT) is a Kleene algebra with an embedded Boolean subalgebra (the complementation operator is defined only on Boolean terms).

Of particular interest is the equational theory of Kleene algebra. Since the Hoare theory of KA (equational implications of the form $r = 0 \rightarrow p = q$), the Hoare theory of KAT, and the equational theory of KAT all reduce to the equational theory of KA, the equational theory of KA suffices to express many interesting properties of programs succinctly. See [1], [4], and [9] for details.

Our first result is the development of the basic theory of disimulations. A disimulation is a relation witnessing the equivalence of two automata. We catalog some of the commonalities of disimulation and the related notion of bisimulation, and show how the former, unlike the latter, can be used as the basis for a complete proof system for the equational theory of KA. This is a significant simplification of the original completeness result of [3].

Our second result is that the production of proofs of KA equations can be automated: there is a *PSPACE* transducer which takes as input equations of Kleene algebra and outputs “algebraic” proofs of them in the proof system described below. The proofs constructed are exponentially long in the worst case, but this is the best that one could expect, unless *PSPACE* = *NP*: deciding the equational theory of KA is a *PSPACE* complete problem [8], so the existence of polynomially long proofs of all equivalences would imply *PSPACE* = *NP*.

This paper is organized as follows. In section 2, we provide the relevant definitions and recall the encoding of finite automata as Kleene algebra terms. In section 3, we develop the basic theory of disimulations and define the proof system. In section 4, we give a *PSPACE* transducer which takes an equation of KA as input and outputs a proof of it. Finally, in section 5, we discuss a companion paper [9] which contains a feasible reduction from the equational theory of KAT to the equational theory of KA.

2 Background

A Kleene algebra is a structure $\mathcal{K} = (K, 0, 1, +, \cdot, *)$ such that $(K, 0, 1, +, \cdot)$ is an idempotent semiring which also satisfies the following laws:

$$\begin{aligned} 1 + a^*a &\leq a^* & 1 + aa^* &\leq a^* \\ b + ax \leq x &\Rightarrow a^*b \leq x & b + xa \leq x &\Rightarrow ba^* \leq x. \end{aligned}$$

The partial order $\leq$ is induced by addition, i.e.,

$$x \leq y \Leftrightarrow x + y = y.$$

A crucial fact is that the set of $n \times n$ matrices over a Kleene algebra has a natural Kleene algebra structure. See [3] for details. At several points in the proof below, we will have to reason about non-square matrices. We would like to know whether the theorems of Kleene algebra hold when the primitive letters are interpreted as matrices of arbitrary dimension and the function symbols are interpreted polymorphically. In general, this is not the case. However, there is a large class of theorems which do survive this treatment, including all theorems used below [6].

2.1 Representing Automata

Matrices over a Kleene algebra are useful because they allow algebraic encodings of automata. Recall the following definitions:

Definition 1. An automaton over a Kleene algebra K is a triple (u, A, v) where u and v are n -dimensional $(0,1)$ -vectors and A is an $n \times n$ matrix over K . The vector u encodes the start states of (u, A, v) and is called the start vector. The vector v encodes the accept states of (u, A, v) and is called the accept vector. The matrix A is called the transition matrix.

Definition 2. The language accepted by (u, A, v) is the element $u^T A^* v$.

Definition 3. The size of (u, A, v) , denoted $|(u, A, v)|$, is the number of states of the automaton, i.e., if A is an $n \times n$ matrix, then $|(u, A, v)| = n$.

This notion of an automaton is more general than necessary for our purposes. Given an alphabet Σ , let $\mathcal{F}_\Sigma$ be the free Kleene algebra on generators Σ ; in the sequel, all automata are over some $\mathcal{F}_\Sigma$. Furthermore, most of the automata we consider have transition matrices whose entries are sums of atomic terms.

Definition 4. Let (u, A, v) be an automaton over $\mathcal{F}_\Sigma$. We say that (u, A, v) is

(a) simple if A can be expressed as a sum

$$A = J + \sum_{a \in \Sigma} a \cdot A_a$$

where J and each A_a is a $(0,1)$ -matrix.

(b) ϵ -free if J is the zero matrix.

(c) deterministic if it is simple, ϵ -free, and u and all rows of each A_a have exactly one 1.

We will make frequent use of automata encoded as algebraic terms. To simplify proofs, we add to the axioms of Kleene algebra four theorems from [3] involving automata. For each theorem we add, it will be clear that the hypotheses of the theorem are easy to check, so proofs constructed using these new rules of inference are verifiable in polynomial time.

The first three theorems listed below are used to construct an automaton accepting the language denoted by a given regular expression (Kleene's Theorem). The lemmas algebraically represent combinatorial constructions on automata. Let (u, A, v) be an automaton accepting γ , (s, B, t) be an automaton accepting δ , and Φ be a sequence of equations or equational implications.

The first theorem is known as the *union lemma*. It represents taking the “disjoint union” of two automata:

$$\frac{\Phi \vdash u^T A^* v = \gamma \quad \Phi \vdash s^T B^* t = \delta}{\Phi \vdash [u \mid s] \left[\begin{array}{c|c} A & 0 \\ \hline 0 & B \end{array} \right]^* \left[\begin{array}{c} v \\ t \end{array} \right] = \gamma + \delta}.$$

The second theorem is known as the *concatenation lemma*. The term vs^T in the upper right corner of the transition matrix in the conclusion represents adding ϵ -transitions from the accept states of (u, A, v) to the start states of (s, B, t) :

$$\frac{\Phi \vdash u^T A^* v = \gamma \quad \Phi \vdash s^T B^* t = \delta}{\Phi \vdash [u \mid 0] \left[\begin{array}{c|c} A & vs^T \\ \hline 0 & B \end{array} \right]^* \left[\begin{array}{c} 0 \\ t \end{array} \right] = \gamma\delta}.$$

The third is known as the *asterate lemma*. The term $A + vu^T$ represents adding ϵ -transitions from the accept states of (u, A, v) back to the start states; we must also add a state to accept the empty word:

$$\frac{\Phi \vdash u^T A^* v = \gamma}{\Phi \vdash [1 \mid u] \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & A + vu^T \end{array} \right]^* \left[\begin{array}{c} 1 \\ v \end{array} \right] = \gamma^*}.$$

The fourth theorem we add allows us to prove that an automaton and the automaton obtained by removing ϵ -transitions are equivalent. Let (u, A, v) and

(u', F, v) be automata of size n , and let J be an $n \times n$ matrix. Suppose that the following equations hold:

$$A = J + A'$$

$$F = A' J^*$$

$$u'^T = u^T J^*.$$

It follows that (u, A, v) and (u', F, v) are equivalent. We add the following theorem to the KA axioms, called the ϵ -elimination lemma:

$$\frac{\Phi \vdash A = J + A' \quad \Phi \vdash F = A' J^* \quad \Phi \vdash u'^T = u^T J^*}{\Phi \vdash u^T A^* v = u'^T F^* v}.$$

In our applications, J is a $(0,1)$ -matrix, so $u^T J^*$ is a $(0,1)$ -vector and F is ϵ -free.

3 The Disimulation Relation

A disimulation (“directed bisimulation”) is a relation witnessing the equivalence of two simple ϵ -free automata. Let (s, B, t) and (u, A, v) be two such automata. Suppose that $|u, A, v| = m$ and $|s, B, t| = n$. Let R be a relation from the states of (s, B, t) to the states of (u, A, v) , and let X be the encoding of R as an $n \times m$ $(0,1)$ -matrix. We say that R is a *disimulation* if the following equations hold:

$$s^T X = u^T \tag{1}$$

$$X A = B X \tag{2}$$

$$X v = t. \tag{3}$$

We call X a *disimulation matrix*. Multiplying X on the right by a characteristic vector of states of (u, A, v) results in a characteristic vector of states of (s, B, t) , hence we call (u, A, v) the *source* automaton and (s, B, t) the *target* automaton. It follows from the axioms of Kleene algebra that the two automata accept the same language [3]. As shown below, disimulations can be used as the basis of a complete proof system for the equational theory of Kleene algebra, unlike the standard notion of bisimulation (recall that equivalent nondeterministic automata may be in different bisimilarity classes [5]). Also cf. “Boolean bisimulations” in [2].

We first note some properties that bisimulations and disimulations share. Recall that the bisimulation relation is an equivalence relation on automata, and that the union of two bisimulations is a bisimulation. Disimulation is a reflexive relation; it is easy to see that the identity matrix satisfies the defining equations of a disimulation.

The composition of two disimulations (with compatible directions) is again a disimulation.

Proposition 1. *Let (u, A, v) , (s, B, t) , and (p, C, q) be automata, with X a disimulation from (u, A, v) to (s, B, t) and Y a disimulation from (s, B, t) to (p, C, q) . Then YX is a disimulation from (u, A, v) to (p, C, q) .*

Proof.

$$\begin{aligned} p^T(YX) &= (p^TY)X = s^TX = u^T \\ (YX)A &= Y(XA) = Y(BX) = (YB)X = (CY)X = C(YX) \\ (YX)v &= Y(Xv) = Yt = q. \end{aligned}$$

It is also the case that the sum of two disimulations is a disimulation.

Proposition 2. *Let (u, A, v) and (s, B, t) be automata, and let X and Y be disimulations from (u, A, v) to (s, B, t) . Then $X + Y$ is a disimulation from (u, A, v) to (s, B, t) .*

Proof.

$$\begin{aligned} s^T(X + Y) &= s^TX + s^TY = u^T + u^T = u^T \\ (X + Y)A &= XA + YA = BX + BY = B(X + Y) \\ (X + Y)v &= Xv + Yv = t + t = t. \end{aligned}$$

We also note that reversing the directions of the transitions and swapping start and accept states of disimilar automata yields automata which are disimilar with the direction of disimulation reversed.

Proposition 3. *Let X be a disimulation from (u, A, v) to (s, B, t) . Then X^T is a disimulation from (t, B^T, s) to (v, A^T, u) .*

Proof. Taking the transpose of the disimulation equations yields

$$\begin{aligned} t^TX^T &= v^T \\ X^TB &= AX^T \\ X^Ts &= u. \end{aligned}$$

Note that the familiar equation

$$(AB)^T = B^TA^T$$

for matrices over a field does not hold for matrices over a Kleene algebra in general, but it does hold if one of the matrices is a $(0,1)$ -matrix.

However, disimulation is not a symmetric relation (hence the “source” and “target” designations). Before demonstrating this, we collect some pairs of automata which are guaranteed to be disimilar.

Proposition 4. *Let (u, A, v) be an automaton and (s, B, t) be the equivalent deterministic automaton obtained from the subset construction. Then (u, A, v) and (s, B, t) are disimilar.*

Proof. This is shown in [3]. The disimulation is the relation which relates a state of (s, B, t) (considered as a set of states of (u, A, v)) to each state of (u, A, v) that it “contains”; the source automaton is (u, A, v) .

Proposition 5. *Let (u, A, v) and (s, B, t) be isomorphic automata. Then (u, A, v) and (s, B, t) are disimilar.*

Proof. Let f be an isomorphism from the states of (s, B, t) to the states of (u, A, v) . Let P be the encoding of f as a permutation matrix. Then

$$A = P^T B P \quad (4)$$

$$u = P^T s \quad (5)$$

$$v = P^T t. \quad (6)$$

Note that only the idempotent semiring axioms are needed to show that

$$P^{-1} = P^T$$

for permutation matrices. Multiplying (4) and (6) on the left by P yields

$$P A = B P$$

$$P v = t.$$

Taking the transpose of (5) yields

$$s^T P = u^T.$$

Therefore P is a disimulation from (u, A, v) to (s, B, t) .

Before proving any more pairs disimilar, we need a lemma. Given a transition matrix M , let δ_M be the transition relation it defines, and let δ_M^a be δ_M restricted to a -transitions for $a \in \Sigma$. Let $\mathcal{A}$ denote the set of states of (u, A, v) , and $\mathcal{B}$ denote the set of states of (s, B, t) .

Lemma 1. *Let (u, A, v) and (s, B, t) be simple, ϵ -free automata, and X a relation from $\mathcal{B}$ to $\mathcal{A}$. Suppose that for each $a \in \Sigma$, $i \in \mathcal{B}$, and $j \in \mathcal{A}$, the “diagram”*

$$\begin{array}{ccc} i & \xrightarrow{X} & \mathcal{A} \\ \downarrow \delta_B^a & & \downarrow \delta_A^a \\ \mathcal{B} & \xrightarrow{X} & j \end{array}$$

commutes, i.e., there is a path from state i to state j above the diagonal if and only if there is a path below the diagonal. Then $XA = BX$.

Proof. We must show that for all i, j , $(XA)_{ij} = (BX)_{ij}$. The commutativity condition implies that for each $a \in \Sigma$, $a \leq (XA)_{ij}$ if and only if $a \leq (BX)_{ij}$. Since (u, A, v) and (s, B, t) are simple, $XA = BX$. Note that because $a + a = a$, it does not matter how many times a occurs in $(XA)_{ij}$ or $(BX)_{ij}$, only whether a occurs.

Proposition 6. *Let (s, B, t) be a deterministic automaton with only accessible states, and let (u, A, v) be the minimal equivalent dfa. Then (u, A, v) and (s, B, t) are disimilar.*

Proof. We say that state i is equivalent (indistinguishable) from state j if and only if for all $w \in \Sigma^*$, $\hat{\delta}(i, w)$ and $\hat{\delta}(j, w)$ are either both accept states or both nonaccept states (i and j are not necessarily states of the same automaton). Let X be a matrix encoding the relation

$$R = \{(i, j) \mid i \in \mathcal{B}, j \in \mathcal{A}, i \text{ and } j \text{ are indistinguishable}\}.$$

Recall that every pair of distinct states of (u, A, v) is distinguishable by minimality. Since (s, B, t) and (u, A, v) are equivalent, the start state of (s, B, t) is related to the start state of (u, A, v) , so $s^T X$ has a 1 in the entry corresponding to the start state of (u, A, v) . To see that the other entries of $s^T X$ are 0, note that each state of (s, B, t) is related to exactly one state of (u, A, v) , by minimality of (u, A, v) . A 1 in an entry of $s^T X$ not corresponding to the start state of (u, A, v) would mean that there is another state of (u, A, v) which is indistinguishable from the start state of (s, B, t) , and thus indistinguishable from the start state of (u, A, v) , contradicting the minimality of (u, A, v) . Therefore $s^T X = u^T$.

The equation $XA = BX$ follows easily from the definition of X and Lemma 1. Finally, we show that the equation $Xv = t$ holds. Let s_A be the start state of (u, A, v) and s_B be the start state of (s, B, t) . Each state in (s, B, t) is accessible, so for any accept state i of (s, B, t) , there is a word w such that $\hat{\delta}_B(s_B, w) = i$. Since (u, A, v) is deterministic and equivalent to (s, B, t) , the state $\hat{\delta}_A(s_A, w)$ must be an accept state and related to i . No nonaccept state of (s, B, t) can be related to an accept state of (u, A, v) , by the definition of X . These considerations imply $Xv = t$.

A similar proof shows that an automaton and the minimal equivalent nfa are disimilar, using properties of the minimal nfa developed in [5].

Proposition 7. *Let (u, A, v) be an automaton. and let (s, B, t) be (u, A, v) with the inaccessible states removed (if (u, A, v) has no accessible states, then $(s, B, t) = (0, 0, 0)$). Then (u, A, v) and (s, B, t) are disimilar.*

Proof. Let X be the matrix encoding the relation from (s, B, t) to (u, A, v) in which a state of (s, B, t) is related to its copy in (u, A, v) . Since start states are by definition accessible, the equation $s^T X = u^T$ holds. Using Lemma 1, it is easy to see that the equation $XA = BX$ holds, and $Xv = t$ holds because t consists of the accessible final states of (u, A, v) .

Since the live states (states with an outgoing path to an accept state) of an automaton are precisely the accessible states of the reverse automaton, Propositions 3 and 7 imply that an automaton and the subautomaton consisting only of live states are also disimilar.

Now, not all equivalent automata are disimilar, just as not all equivalent automata are bisimilar. There do exist disimilar automata which are not bisimilar; in general an automaton and its determinization are not bisimilar. There are also bisimilar automata which are not disimilar. Consider the deterministic automata

$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & a & 0 \\ 0 & 0 & a \\ a & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right), \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ a & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right).$$

Both automata accept the language a^* , but neither system of equations has a solution. That is, it is impossible to solve for a 2×3 or 3×2 disimulation matrix X . Recall, however, that any two equivalent deterministic automata are bisimilar.

Continuing with this example, let us call the three-state automaton (s_1, D_1, t_1) and the two-state automaton (s_2, D_2, t_2) . Let (p, M, q) denote the (minimal) one-state automaton accepting a^* . Then disimulations exist in the indicated directions:

$$(s_1, D_1, t_1) \leftarrow (p, M, q) \rightarrow (s_2, D_2, t_2).$$

If the directions could be reversed at will, then the two disimulations could be made to point in the same direction. They could then be composed, which would yield a disimulation from (s_1, D_1, t_1) to (s_2, D_2, t_2) , which is impossible. Therefore disimulation is not a symmetric relation. However, by the above propositions, it is always the case that two equivalent ϵ -free automata (u_1, A_1, v_1) and (u_2, A_2, v_2) can be proven equivalent using the automata and disimulations

$$(u_1, A_1, v_1) \rightarrow \text{accessible dfa} \leftarrow \text{minimal dfa} \rightarrow \text{accessible dfa} \leftarrow (u_2, A_2, v_2).$$

Here “accessible dfa” refers to the dfa obtained by the standard subset construction, with the inaccessible states removed (the dfa with the inaccessible states is the “full dfa”).

Since disimulations are in general not symmetric, the intermediate automata in the above proof cannot necessarily be “composed away” by reversing directions where appropriate. Note that if this were possible, then any two equivalent automata would have a polynomial-sized disimulation witnessing their equality. This would imply $PSPACE = P$, since disimulations can be constructed using a modification of the standard table-filling (polynomial-time) algorithm to compute bisimulations.

We now define our proof system. Given α and β , two equivalent KA terms, a proof that $\alpha = \beta$ consists of:

1. simple, ϵ -free automata (u_1, A_1, v_1) , (u_n, A_n, v_n) , and proofs from the KA axioms that $\alpha = u_1^T A_1^* v_1$, $\beta = u_n^T A_n^* v_n$.

2. A sequence

$$(u_1, A_1, v_1), X_1, (u_2, A_2, v_2), X_2, \dots, X_{n-1}, (u_n, A_n, v_n)$$

where (u_i, A_i, v_i) is a simple, ϵ -free automaton and X_i is a disimulation matrix between (u_i, A_i, v_i) and $(u_{i+1}, A_{i+1}, v_{i+1})$, along with a tag indicating the source automaton.

The above considerations show completeness of this proof system (assuming we can generate a simple, ϵ -free automaton for each term, which is shown below). It is easy to see that such a proof can be verified in polynomial time.

4 Proving KA Equations

In this section, we give an algorithm to generate proofs and show that it can be implemented by a *PSPACE* transducer. Given a KA term α , let $|\alpha|$ be the number of nodes in the syntax tree of α .

Theorem 1. *Let $\alpha = \beta$ be an equation of Kleene algebra. A proof that $\alpha = \beta$ can be produced by a transducer using only polynomially many (in $|\alpha| + |\beta|$) worktape cells.*

Respecting the space bound is nontrivial; we require several terms of exponential size, some of which are constructed from terms which are themselves exponentially large. To simplify proving that the space bound is not violated, we divide the construction of the proof into stages. For each stage, we show that both the terms and the proofs required at that stage can be constructed in *PSPACE*. The stages:

1. Construct an nfa accepting α , an nfa accepting β , and proofs thereof.
2. For each nfa, construct an equivalent ϵ -free nfa, and an equivalence proof.
3. For each ϵ -free nfa, construct an equivalent accessible dfa, and a disimulation matrix between them.
4. Construct the minimal dfa equivalent to the accessible dfa accepting α , and a disimulation matrix between them.
5. Construct the disimulation matrix between the minimal dfa for α and the accessible dfa for β .

Stages 2 through 5 require one or more terms from previous stages. We treat each stage independently, and show that there are transducers which generate the required terms and/or proofs at each stage. To combine all of the stages, we use the following fact about the composition of space-bounded transducers.

Lemma 2. *Suppose $f(x)$ can be computed by a *PSPACE* transducer F , and $g(x)$ can be computed by a *PLSPACE* transducer G (a transducer using polylog many worktape cells in the size of its input). Then $g(f(x))$ can be computed by a *PSPACE* transducer.*

Proof. Note that $|f(x)|$ might be exponential in $|x|$, so there is not necessarily enough space to write down $f(x)$ in its entirety. Rather, a *PSPACE* transducer H computing $g(f(x))$ computes $f(x)$ on a demand-driven basis. On input x , H begins by running G . Whenever a bit of $f(x)$ is needed, H saves the current state of G and begins running F on input x , disregarding the output of F until the required bit of $f(x)$ is produced. It then resumes running G , supplying the requested bit of $f(x)$. The transducer H needs polynomially many worktape cells to run F , polynomially many cells to count up to the length of $f(x)$, and polynomially many cells for G 's worktape, since G needs at most $O((\log |f(x)|)^d) \leq O(|x|^m)$ for some m .

4.1 Stage 1: Regular Expression to Automaton

We first show that the inductive construction used in the proof of Kleene's theorem can be performed by a *PSPACE* machine. Given a term α , the machine must construct an automaton (u, A, v) accepting α , and a proof that $u^T A^* v = \alpha$.

Given $a \in \Sigma$, the following automaton accepts the language $\{a\}$:

$$\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right).$$

There are also one-state automata for $\emptyset$ and ϵ : $([0], [0], [0])$ and $([1], [1], [1])$, respectively. We assume that for every $a \in \Sigma$, the machine has a proof that

$$a = [1 \ 0] \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

stored in its finite control. We also assume that the machine can output proofs of the equations

$$0 = 00^*0$$

$$1 = 11^*1.$$

For the inductive step, the machine can work its way up the syntax tree of α , constructing automata as dictated by the union, concatenation, and asterate lemmas. At each step, it outputs the appropriate equation, i.e., the conclusion of one of the three lemmas. When finished, the machine will have constructed an automaton accepting α and also will have printed a proof of this fact on the output tape. All of the terms appearing in the proof are polynomial in the size of α and straightforward to construct.

4.2 Stage 2: Automaton to ϵ -Free Automaton

We now show that there is a transducer which takes a simple automaton (u, A, v) as input and constructs from it an equivalent simple ϵ -free automaton (u', F, v) , and that there is a transducer which takes as input the pair $((u, A, v), (u', F, v))$ and outputs a proof of the equivalence.

Constructing the ϵ -free automaton, (u', F, v) , is easy. Since (u, A, v) is simple,

$$A = J + \sum_{a \in \Sigma} a \cdot A_a.$$

as in Definition 4.(a). The transducer computes J from (u, A, v) and then computes J^* , which is just the reflexive transitive closure of the relation denoted by J . It also computes

$$A' = \sum_{a \in \Sigma} a \cdot A_a.$$

Then

$$u'^T = u^T J^*$$

$$F = A' J^*.$$

It is easy to see that both u' and F can be constructed in *PSPACE*. Note that (u', F, v) might not be simple, but can easily be made so using additive idempotence. To prove equivalence, the proof-generating transducer uses the ϵ -elimination lemma. It must prove the following hypotheses:

$$A = J + A'$$

$$F = A' J^*$$

$$u'^T = u^T J^*$$

all of which are easily proven in *PSPACE*. The machine must also prove that the term J^* is the star of J . First, the machine proves

$$1 + J(1 + J + J^2 + \cdots + J^n) \leq (1 + J + J^2 + \cdots + J^n)$$

by direct computation. This inequality is true; if the i, j entry of JJ^n is 1, then there is a path of length $n + 1$ from i to j (viewing J as the adjacency matrix of a graph). Since J has only n vertices, this path must repeat at least one vertex, and so there will be a 1 in the i, j entry of J^k for some $k < n + 1$. Reasoning in KA,

$$J^* \leq 1 + J + J^2 + \cdots + J^n.$$

Next, the machine generates a proof that for any x ,

$$1 + x + x^2 + \cdots + x^n \leq x^*.$$

This inequality is an easy consequence of the KA axioms. Substituting J for X and combining these two inequalities yields

$$1 + J + J^2 + \cdots + J^n = J^*.$$

4.3 Stage 3: ϵ -Free Automaton to Deterministic Automaton

It must now be shown that there is a *PSPACE* transducer which takes in (u', F, v) , a simple ϵ -free automaton, and outputs (s, D, t) , an equivalent accessible deterministic automaton. Let $|(u', F, v)| = n$. To generate (s, D, t) , the machine performs the standard subset construction on (u', F, v) , with the added condition that it tests each subset for accessibility before granting it state status. The following lemma verifies that this test can be performed in *PSPACE*.

Lemma 3. *Let (u', F, v) be a simple ϵ -free automaton with n states. It is decidable in $O(n^2)$ space whether C , a set of states of (u, F, v) , is accessible when considered as a state in the deterministic automaton obtained from (u', F, v) by the subset construction.*

Proof. We first give a nondeterministic linear space machine. The machine starts with (u', F, v) and the characteristic vector of C written on its input tape. It begins by writing the start vector u' on its worktape. If $u' = C$, it halts and answers yes. Otherwise it guesses an $a \in \Sigma$ and overwrites its worktape contents with the characteristic vector of $\delta_F(u', a)$. If this is equal to C , it accepts, otherwise it guesses another letter and repeats. At any time, the machine must store only $O(n)$ bits of information. By Savitch's theorem, there is an equivalent deterministic machine running in $O(n^2)$ space.

To construct s , the machine counts from 0 to $2^n - 1$ in binary (each number is identified with a subset of states of (u', F, v) by treating its binary representation as a characteristic vector). For each i between 0 and $2^n - 1$, it tests whether i represents an accessible state. If i does not, the machine proceeds to the next i . If i does represent an accessible state, the machine outputs 1 if i represents precisely the set of start states of (u', F, v) , and 0 otherwise. The construction of t is similar, except the machine outputs 1 if any state in the subset represented by i is an accept state, or 0 if none are.

The construction of D , the transition matrix, requires three counters. The first two, i and j , range from 0 to $2^n - 1$, and are used to keep track of the rows and columns of D , respectively. The third counter, c , ranges from 0 to $m - 1$, where $m = |\Sigma|$. The machine starts with all counters set to zero. It begins by testing i for accessibility. If i is inaccessible, it increments i and repeats. If i does correspond to an accessible state, it then tests each possible value of j for accessibility. If j is not accessible, it increments j . If j does represent an accessible state, it tests each $a_k \in \Sigma$ to determine whether $\delta'_F(i, a_k) = j$. If yes, it outputs a_k . If none of the a_k tests succeed, it outputs 0. After testing all of the a_i 's, the machine resets c to 0 and goes to the next j . After checking all of the j 's, the machine resets j to 0 and goes to the next i .

This transducer runs in $O(n^2)$ space, where n is $|(u', F, v)|$. The machine requires $O(n^2)$ space to perform the test in Lemma 2 and a few counters which range up to $2^n - 1$.

Let d be $|(s, D, t)|$ and let X be the $d \times n$ matrix encoding the relation in which a state of (s, D, t) is related to all of the states of (u, F, v) that it "contains".

Note that this is the composition of the disimulation between (s, D, t) and the full dfa with the disimulation between the full dfa and (u', F, v) .

We must show that the disimulation matrix can be computed without violating the space bound. The transducer which takes the pair $((u', F, v), (s, D, t))$ and outputs the disimulation matrix can use only polynomially many (in $|u', F, v|$) cells, although $|s, D, t|$ may be exponential in n . To construct X , the machine needs one counter ranging from 0 to $2^n - 1$. For each i between 0 and $2^n - 1$, the machine tests the subset of states encoded by i for accessibility. If it is accessible, it outputs the binary representation of i as a row vector. If i does not represent an accessible state, it goes to $i + 1$.

4.4 Stage 4: Deterministic Automaton to Minimal Deterministic Automaton

At this stage, we require two transducers. The first constructs the minimal deterministic automaton equivalent to a given accessible deterministic automaton, and the second takes as input a pair (dfa, equivalent minimal dfa) and outputs the disimulation matrix between them. The minimal dfa (p, M, q) is constructed by examining (s, D, t) and outputting the least-numbered state in each equivalence class of a Myhill-Nerode relation. We require a lemma establishing a space bound on the procedure to identify equivalent states.

Lemma 4. *Let (s, D, t) be a deterministic automaton. It is decidable in polylog space whether i and j , two states of (s, D, t) , are equivalent.*

Proof. We first give an *NLOGSPACE* procedure to recognize distinguishable (inequivalent) states. The machine begins with (s, D, t) , i , and j written on its input tape. If one of i, j is an accept state and the other is not, the machine halts and answers distinguishable. Otherwise it guesses an $a_1 \in \Sigma$ and overwrites its worktape contents with $\delta_D(i, a_1)$ and $\delta_D(j, a_1)$. If exactly one of these states is an accept state, the machine halts and answers distinguishable. If not, it guesses an $a_2 \in \Sigma$ and repeats the procedure. At any time, the machine has to remember only two states of (s, D, t) , and so it runs in *NLOGSPACE*. By Savitch's theorem, there is an equivalent deterministic machine running in $O((\log |s, D, t|)^2)$ space.

To construct p , the start vector, the machine scans s . For each state i , it checks whether i is equivalent to some lower-numbered state. If yes, it skips to the next i . If i is the least-numbered state in its equivalence class, the machine outputs a 1 if i is equivalent to the start state of (s, D, t) , and 0 otherwise. The accept vector, q , is constructed similarly. The machine scans through t , and for each state i that is the least-numbered state in its equivalence class, it outputs 1 if i is an accept state, 0 if i is not.

The construction of the transition matrix M resembles the construction of the transition matrix of the deterministic automaton in the previous stage. The machine maintains two counters, i and j . It scans through the states of (s, D, t) , and for each state i which is the least-numbered state in its equivalence class, it

tests each state j in turn, outputting D_{ij} for each j which is the first state in its equivalence class. It is easy to see that this procedure can be done in *PLSPACE* and does indeed generate the equivalent minimal dfa.

A transducer to construct the disimulation matrix X from the pair $((s, D, t), (p, M, q))$ uses a straightforward modification of Lemma 4 to generate X in *PLSPACE*. By Lemma 2, the above terms can be generated in *PSPACE*.

4.5 Stage 5: DFA for β Disimilar to Minimal Automaton for α

It suffices to use the procedure from the previous stage to generate the disimulation matrix between the two automata.

5 KAT Equations

In [4], it is shown that the equational theory of Kleene algebra with tests reduces to the equational theory of Kleene algebra. The Hoare theory of KAT also reduces to the equational theory of KAT. In [9], we show that these reduction can be done feasibly. Note that the Hoare theory of KAT suffices to encode Propositional Hoare Logic [7], which means that many interesting properties of programs can ultimately be expressed as equations of Kleene algebra.

6 Conclusion

We have introduced the notion of *disimulation*, and shown how many common constructions which produce an equivalent automaton from a given automaton (e.g. determinization, minimization, removal of dead/live states) yield disimilar automata. We have also shown that disimulation, when combined with Kleene's theorem and basic facts about reflexive transitive closures (used for ϵ -elimination) yields a complete proof system for the equational theory of Kleene algebra, and that these proofs can be constructed by a *PSPACE* transducer. The proofs are exponentially long in the worst case; identifying interesting classes of equations with short proofs and/or better proof search strategies remains to be done. We remark that using the reduction of the equational theory of KAT to the equational theory of KA mentioned above, it is possible to produce polynomially long proofs of deterministic **while** program equivalence [9].

Acknowledgments

I would like to thank Dexter Kozen for many helpful comments and informative conversations, and the anonymous RelMiCS referees for many valuable suggestions. This material is based upon work supported by the National Science Foundation under Grant No. 0635028.

References

- [1] Cohen, E.: Hypotheses in Kleene Algebra. Technical Report TM-ARH-023814, Bellcore (1993), <http://citeseer.ist.psu.edu/1688.html>
- [2] Fitting, M.: Bisimulations and Boolean Vectors. *Advances in Modal Logic* 4, 97–125 (2003)
- [3] Kozen, D.: A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Infor. and Comput* 110(2), 366–390 (1994)
- [4] Kozen, D., Smith, F.: Kleene Algebra with Tests: Completeness and Decidability. In: van Dalen, D., Bezem, M. (eds.) *CSL 1996. LNCS*, vol. 1258, pp. 224–259. Springer, Heidelberg (1997)
- [5] Kozen, D.: Automata and Computability. In: *Undergraduate Texts in Computer Science*. Springer, Heidelberg (1997)
- [6] Kozen, D.: Typed Kleene Algebra. Technical Report 98-1669, Computer Science Department, Cornell University (March 1998)
- [7] Kozen, D.: On Hoare Logic and Kleene Algebra with Tests. *Trans. Computational Logic* 1(1), 60–76 (2000)
- [8] Stockmeyer, L.J., Meyer, A.R.: Word Problems Requiring Exponential Time. In: *Proc. 5th Symp. Theory of Computing*, pp. 1–9 (1973)
- [9] Worthington, J.: Feasibly Reducing KAT Equations to KA Equations, <http://arxiv.org/abs/0801.2368>

Author Index

- Balbiani, Philippe 4
Berghammer, Rudolf 22
Bolduc, Claude 289
Braßel, Bernd 37
- Christiansen, Jan 37
- De Carufel, Jean-Lou 54, 69
Desharnais, Jules 54, 69
Diedrich, Florian 84
Düntsch, Ivo 99
- Furusawa, Hitoshi 110
- Griffin, Timothy G. 123
Gurney, Alexander J.T. 123
Guttmann, Walter 138
- Höfner, Peter 191, 206
Honda, Kazumasa 221
Hopkins, Mark 155, 173
- Ishida, Toshikazu 221
- Jipsen, Peter 234
- Kahl, Wolfram 243
Kawahara, Yasuo 221, 259, 274
Kehden, Britta 22, 84
Ktari, Béchir 289
- Lajeunesse-Robert, François 289
- Meinicke, L.A. 304
Meseguer, José 337
Möller, Bernhard 320
- Neumann, Frank 84
Nishizawa, Koki 110
- Pauly, Marc 1
- Rocha, Camilo 337
- Schmidt, Gunther 3, 352
Solín, K. 304
Struth, Georg 206, 234
- Tinchev, Tinko 4
Tsumagari, Norihiro 110
- Winter, Michael 99, 274, 367
Worthington, James 382