6th Workshop on Fixed Points in Computer Science

FICS 2009

Coimbra, Portugal, 12-13 September 2009

Proceedings

TTÜ KÜBERNEETIKA INSTITUUT
Institute of Cybernetics at TUT

6th Workshop on Fixed Points in Computer Science

FICS 2009

Coimbra, Portugal, 12–13 September 2009

Proceedings

Institute of Cybernetics at Tallinn University of Technology

Tallinn ○ 2009

6th Workshop on Fixed Points in Computer Science
FICS 2009
Coimbra, Portugal, 12–13 September 2009
Proceedings

Edited by Ralph Matthes and Tarmo Uustalu

Institute of Cybernetics at Tallinn University of Technology
Akadeemia tee 21, EE-12618 Tallinn, Estonia
`http://www.ioc.ee/`

ISBN 978-9949-430-29-1

Printed by Alfapress

Preface

This volume is the proceedings of the *6th Workshop on Fixed Points in Computer Science, FICS 2009*, to take place in Coimbra, Portugal, 12–13 September 2009. This workshop will be held as a satellite workshop of the *23rd International Workshop on Computer Science Logic*, the *18th EACSL Annual Conference, CSL 2009* (7–11 September), in colocation with the *11th ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP 2009* (7–9 September), and the *19th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2009* (9–11 September).

FICS is intended as a forum for researchers to present their results to those members of the computer science and logic communities who study or apply the theory of fixed points. FICS was initiated by Zoltán Ésik. Previous workshops of the series were held in Brno (1998, MFCS/CSL workshop), Paris (2000, LC workshop), Florence (2001, PLI workshop), Copenhagen (2002, LICS (FLoC) workshop), Warsaw (2003, ETAPS workshop).

The two-day programme of FICS 2009 consists of three invited and 15 contributed talks, represented in this volume by short resp. extended abstracts. Our invited speakers are Robin Cockett (University of Calgary), Javier Esparza (Technische Universität München) and Yde Venema (Universiteit van Amsterdam). The contributed talks were selected from among 22 submissions by an international program committee. Each submission was reviewed by three PC members or additional referees. But as the selection was based on extended abstracts, not full papers, the nature of these proceedings is informal. A formal post-proceedings, consisting of revised, full versions of selected contributions, will be published as a special issue of the journal *Theoretical Informatics and Applications* of EDP Sciences.

We are grateful to Zoltán Ésik for entrusting us to organize this edition of FICS, to our PC and additional referees for the work they did on the submissions to the workshop and have yet to do on the special issue, to our invited speakers for accepting to come and talk in Coimbra, and to the authors of all submissions for choosing FICS. We are indebted to the organizing committee of CSL/PPDP/LOPSTR 2009, in particular, Reinhard Kahle, José Carlos Espírito Santo and Pedro Quaresma, for offering to host FICS 2009 and for taking care of all practical arrangements concerning the workshop. To Christian Choffrut, we extend our thanks for agreeing to publish the special issue, and to Andrei Voronkov, for powering us with EasyChair.

FICS 2009 is sponsored by EXCS, the Estonian Centre of Excellence in Computer Science, funded mainly by the European Regional Development Fund.

Ralph Matthes and Tarmo Uustalu

Toulouse—Tallinn, 24 August 2009

Organization

Programme Committee

Yves Bertot (INRIA Sophia Antipolis)
Anuj Dawar (University of Cambridge)
Peter Dybjer (Chalmers University of Technology)
Zoltán Ésik (University of Szeged)
Masahito Hasegawa (Kyoto University)
Anna Ingólfsdóttir (Reykjavík University)
Ralph Matthes (IRIT, Toulouse) (co-chair)
Jan Rutten (CWI and Vrije Universiteit Amsterdam)
Luigi Santocanale (LIF, Marseille)
Alex Simpson (University of Edinburgh)
Tarmo Uustalu (Institute of Cybernetics, Tallinn) (co-chair)
Igor Wałukiewicz (LaBRI, Bordeaux)

Additional Referees

Luca Aceto, Ichiro Hasuo, Shin-ya Katsumata, Kenshi Miyabe, Milad Niqui, Joshua Sack, Alexandra Silva, Frank Stephan, Eijiro Sumii

Organizing Committee

Ana Almeida (co-chair), Sabine Broda, José Carlos Espírito Santo, Mário Florido, Gonçalo Gutierres, Reinhard Kahle (co-chair), Isabel Oitavem, Pedro Quaresma, João Rasga, Carlota Simões

Host Institution

Departamento de Matemática, Universidade de Coimbra

Sponsor

Estonian Centre of Excellence in Computer Science, EXCS
(funded mainly by the European Regional Development Fund)

Euroopa Liit
Euroopa
Regionaalarengu Fond

Eesti tuleviku heaks

Table of Contents

Invited Talks

Contributed Talks

Pola: A Language for PTIME Programming

Michael J. Burrell
Computer Science Department, University of Western Ontario,
London, Ontario N6A 5B7, Canada
Robin Cockett and Brian F. Redmond
Department of Computer Science, University of Calgary,
Calgary, Alberta T2N 1N4, Canada
robin@cpsc.ucalgary.ca

Pola is a functional style programming language—currently under development—whose type system guarantees that all its programs run in polynomial time. Indeed, as every polynomial time (PTIME) algorithm can be rendered in Pola, the language is complete for polynomial time programming. Furthermore, to someone familiar with functional programming, Pola enforces a style of programming which is—relative to what is being achieved—quite natural.

Pola was inspired by the realization that Bellantoni and Cook's system of safe recursion [1] can be viewed as the proof theory of a polarized logic [4]. As polarized logic was developed to model games, Pola has inherited some game theoretic terminology. In particular, rather than having "safe" and "normal" types, Pola has "player" and "opponent" types. The game theoretic view is that the opponent drives the iteration of computation while the player responds in constant time (and space) in the context he is given. This semantics has a particularly appealing and simple categorical presentation using a fibration whose fibers are affine categories together with a notion of "comprehended" inductive fixed points.

The development of *implicit* systems for PTIME programs has considerable history. In particular, Bellantoni and Cook's system of safe recursion is a simplification of a slightly earlier system of Leivant [8]. That system used tiered recursion and supported a general class of inductive data. Bellantoni and Cook, besides simplifying the system of tiers by safe recursion, also abandoned general inductive data in favor of modeling binary numbers: this limits their system as a basis for programming. Hofmann, in his habilitationsschrift [6], aware of all the above, developed a modal type system for PTIME programs which he modeled in a presheaf topos. In particular, Hofmann suggested that constant time affine computations could be used as the basis for stepping up to polynomial time. Pola's player world (in a given opponent context) is, as mentioned above, both affine and populated by constant time computations. Furthermore, in Pola these player worlds can additionally support arbitrary coinductive fixed points.

An important step in the development of Pola was to provide uniform recursion schemes both for inductive and coinductive data. Pola's recursion scheme derives from the circular proof systems of Luigi Santocanale [11], which also appeared earlier (with other schemes) in Varmo Vene's thesis [12]. This recursion scheme, as used in Pola, is interesting as it has some built-in higher-order content which allows one to avoid Colson's objection [5, 10] to first-order recursion schemes and to express recursive programs in a reasonably natural manner.

The original motivation behind Pola was driven by the investigation of implicit type systems for low complexity programs. However, the fact that there is a relatively simple categorical semantics for this system, suggests another important direction: perhaps, categorical techniques can be usefully employed to obtain a deeper structural understanding of PTIME?

References

[1] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

[2] V.-H. Caseiro. *Equations for Defining Poly-time Functions*. PhD thesis, University of Oslo, 1997.

[3] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, ed., *Proc. of 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, pp. 24–30. North Holland, 1964.

[4] R. Cockett and R. Seely. Polarized category theory, modules and game semantics. *Theory and Appl. of Categ.*, 18:4–101, 2007.

[5] L. Colson. About primitive recursive algorithms. *Theor. Comput. Sci.*, 83(1):57–69, 1991.

[6] M. Hofmann. *Type Systems for Polynomial-Time Computation*. Habilitation thesis, University of Darmstadt, 1999.

[7] M. Hofmann. Linear types and non-size-increasing polynomial time computation. *Inform. and Comput.*, 183(1):57–85, 2003.

[8] D. Leivant. Stratified functional programs and computational complexity. In *Conf. Record of 20th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '93 (Charleston, SC, Jan. 1993)*, pp. 325–333. ACM Press, 1993.

[9] D. Leivant and J.-Y. Marion. Ramified recurrence and computational complexity II: substitution and polyspace. In L. Pacholski and J. Tiuryn, eds., *Selected Papers from 8th Int. Wksh. on Computer Science Logic, CSL '94 (Kazimierz, Sept. 1994)*, v. 933 of *Lect. Notes in Comput. Sci.*, pp. 486–500. Springer, 1995.

[10] Y. N. Moschovakis. On Colson's theorem. Invited talk at 2nd Panhellenic Logic Symp. (Delphi, July 1999). Available at `http://www.mathphys.ntua.gr/logic/symposium/articles/moschova.ps`.

[11] L. Santocanale. A calculus of circular proofs and its categorical semantics. In M. Nielsen and U. Engberg, eds., *Proc. of 5th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2002 (Grenoble, Apr. 2002)*, v. 2303 of *Lect. Notes in Comput. Sci.*, pp. 357–371. Springer, 2002.

[12] V. Vene. *Categorical Programming with Inductive and Coinductive Types*. PhD thesis, University of Tartu, 2000.

Solving Fixed-Point Equations on ω-Continuous Semirings

Javier Esparza, Stefan Kiefer, and Michael Luttenberger
Institut für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748 München, Germany
esparza@in.tum.de

In the talk I will survey several results we have recently obtained on solving equations $X = f(X)$, where $f(X)$ is a polynomial over a semiring. I will sketch a generalization of Newton's method, and, time permitting, a technique called derivation tree analysis.

Fixpoint Logics and Automata: A Coalgebraic Approach

Yde Venema
Institute for Logic, Language and Computation, Universiteit van Amsterdam,
Plantage Muidergracht 24, NL-1018 TV Amsterdam, The Netherlands
y.venema@uva.nl

A long and fertile tradition in theoretical computer science, going back to the work of Büchi and Rabin, links the field of (fixpoint) logic to that of automata theory. In particular, automata operating on potentially infinite structures such as streams, trees, graphs or transition systems, provide an invaluable tool for the specification and verification of the ongoing behavior of systems. An interesting phenomenon in this branch of automata theory is that some of key results (such as determinization) hold for stream automata only, while many others hold of stream, tree and graph automata alike, and can even be proved for automata operating on yet more complex structures. This naturally begs the question whether the theory of automata operating on infinite objects can be lifted a higher level of generality.

The aim of the talk is to advocate Coalgebra as a nice framework for the development of such a theory. The basic observation is that streams, trees and transition systems are all examples of coalgebras of a certain type. In general, a coalgebra consists of a pair consisting of a set S of states together with a transition map from S to the set FS—here F is the type of the coalgebra, given as a functor F on the category Set (with sets as objects and functions as arrows). Universal Coalgebra is the emerging mathematical theory of such state-based evolving systems, in which concepts such as behavior, indistinguishability, invariants, etc can be modelled in a natural way.

In the talk we give a quick introduction to coalgebra, and we introduce various kinds of automata that are supposed to operate on coalgebras, generalizing the well-known automata that operate on streams, trees, etc. The criterion under which such an automaton accepts or rejects a pointed coalgebra is formulated in terms of a two-player parity game, and with each kind of coalgebra automaton we may naturally associate a language of coalgebraic fixpoint logic. Concretely, we show that some of the central results in automata theory can be generalized to the abstraction level of coalgebras. As examples of such results, we will see that the class of recognizable languages of coalgebras is closed under taking unions, intersections, projections, and complementation. We also prove that if a coalgebra automaton accepts some coalgebra it accepts a finite one of bounded size. Many of these results are based on an explicit construction which transforms a given alternating F-automaton into an equivalent nondeterministic one, of bounded size. Finally, we compare various notions of coalgebra automata, and discuss the foundations of a universal theory of automata.

The point behind the introduction of automata at this level of abstraction is that, in the spirit of Universal Coalgebra, we may gain a deeper understanding of automata theory by studying properties of automata in a uniform manner, parametric in the type of the recognized structures.

On Core XPath with Inflationary Fixed Points

Loredana Afanasiev
Informatics Institute, University of Amsterdam,
Science Park 107, NL-1098 XG Amsterdam, The Netherlands
lafanasi@science.uva.nl
Balder ten Cate
INRIA Saclay - Île-de-France and ENS de Cachan,
61 avenue du President Wilson, F-94235 Cachan Cedex, France
balder.tencate@gmail.com

Abstract

In this report, we prove the undecidability of Core XPath 1.0 (CXP) [6] extended with an *Inflationary Fixed Point (IFP)* operator. We prove that the satisfiability problem of this language is undecidable. In fact, the fragment of CXP+IFP containing only the self and descendant axes is already undecidable.

1 Introduction

In [1], an extension of the XML query language XQuery with an inflationary fixed point operator was proposed and studied. The motivation for this study stems from practical use cases. The existing mechanism in XQuery for expressive recursive queries (i.e., user defined recursive functions) is procedural in nature, which makes queries both hard to write and hard to optimize. The inflationary fixed point operator provides a declarative means to specify recursive queries, and is more amenable to query optimization since it blends in naturally with algebra-based query optimization frameworks such as the one of MonetDB/XQuery [3]. Indeed, it was shown in [1] that a significant performance gain can be achieved in this way.

While the empirical evidence is there, a foundational question remains: how feasible it is to do static analysis for recursive queries specified by means of the fixed point operator. Specifically, are there substantial fragments of XQuery with the fixed point operator for which static analysis tasks such as satisfiability are decidable?

In this paper we give a strong negative answer. Our main result states that, already for the downward-looking fragment of Core XPath 1.0 with the inflationary fixed point operator (CXP+IFP), satisfiability is undecidable. The proof is based on a reduction from the undecidable halting problem for 2-register machines (cf. [4]), and borrows ideas from the work of Dawar et al. [5] on the Modal Iteration Calculus (MIC), an extension of modal logic with inflationary fixed points.

A second question we address in this paper is the relationship between CXP+IFP and MIC. While similar in spirit, it turns out that the two formalisms differ in subtle ways. Nevertheless, we obtain a translation from 1MIC (the fragment of MIC that does not involve simultaneous induction) to CXP+IFP node expressions.

In [5], after showing that the satisfiability problem for MIC on arbitrary structures is highly undecidable, the authors ask whether there are still useful fragments, and also whether the logic has any relevance for practical applications. Our results shed some light on these questions. We obtain as a part of our investigation that the satisfiability problem for 1MIC is already undecidable on finite trees, and the relationship between MIC and CXP+IFP adds relevance to the study of MIC.

2 Preliminaries

2.1 Core XPath 1.0 Extended with IFP (CXP+IFP)

Core XPath 1.0 (CXP) was introduced in [6] to capture the navigational core of XPath 1.0. The definition that we use here differs slightly from the one of [6]. We consider only the downward axes *child* and *descendant* (plus the *self* axis), both in order to facilitate the comparison with MIC, and because this will suffice already for our undecidability result. We will briefly comment on the other axes later. Other differences with [6] are that we allow filters and union to be applied to any expressions.

We consider the extension of CXP, which we call CXP+IFP, with an inflationary fixed-point operator. This inflationary fixed-point operator was first proposed in [1] in the context of XQuery, and is here naturally adapted to the setting of CXP. We first give the syntax and semantics of CXP+IFP, and then discuss the intuition behind the operator.

Definition 2.1. Syntax and Semantics of CXP+IFP
Let Σ be a set of labels and VAR a set of variables. The CXP+IFP expressions are defined as follows:

$$\begin{array}{rcl} axis & ::= & \mathsf{self} \mid \mathsf{child} \mid \mathsf{desc} \\ step & ::= & axis::l \mid axis::{}^{*} \\ \alpha & ::= & step \mid \alpha_1/\alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \;\mid \alpha[\varphi] \mid X \mid \mathsf{with}\ X\ \mathsf{in}\ \alpha_1\ \mathsf{recurse}\ \alpha_2 \\ \varphi & ::= & \mathsf{false} \mid \langle\alpha\rangle \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X \end{array}$$

where $l \in \Sigma$ and $X \in VAR$. The α expressions are called *path expressions*, the φ expressions are called *node expressions*. The $\mathsf{with} \ldots \mathsf{in} \ldots \mathsf{recurse} \ldots$ operator is called the WITH operator, while X, α_1, and α_2 in the expression $\mathsf{with}\ X\ \mathsf{in}\ \alpha_1\ \mathsf{recurse}\ \alpha_2$ are called the *variable*, the *seed*, and the *body* of the recursion.

The CXP+IFP expressions are evaluated on *finite node-labeled trees*. Let $T = (N,R,L)$ be a finite node-labeled tree, where N is a finite set of nodes, $R \subset N \times N$ is the child relation in the tree, and L is a function from N to a set of labels. Let $g(\cdot)$ be an assignment function from variables to sets nodes, $g : VAR \to \wp(N)$. Then the semantics of CXP+IFP expressions are as follows:

$$\begin{array}{rcl} \llbracket \mathsf{self} \rrbracket_{T,g} & = & \{(u,u) \mid u \in N\} \\ \llbracket \mathsf{child} \rrbracket_{T,g} & = & R \\ \llbracket axis::l \rrbracket_{T,g} & = & \{(u,v) \in \llbracket axis \rrbracket_T \mid L(u) = l\} \\ \llbracket axis::{}^{*} \rrbracket_{T,g} & = & \llbracket axis \rrbracket_T \\ & & \\ \llbracket \alpha_1/\alpha_2 \rrbracket_{T,g} & = & \{(u,v) \mid \exists w.(u,w) \in \llbracket \alpha_1 \rrbracket_{T,g} \wedge (w,v) \in \llbracket \alpha_2 \rrbracket_{T,g}\} \\ \llbracket \alpha_1 \cup \alpha_2 \rrbracket_{T,g} & = & \llbracket \alpha_1 \rrbracket_{T,g} \cup \llbracket \alpha_2 \rrbracket_{T,g} \\ \llbracket \alpha[\varphi] \rrbracket_{T,g} & = & \{(u,v) \in \llbracket \alpha \rrbracket_{T,g} \mid v \in \llbracket \varphi \rrbracket_{T,g}\} \\ \llbracket X \rrbracket_{T,g} & = & N \times g(X), X \in VAR \\ \llbracket \mathsf{with}\ X\ \mathsf{in}\ \alpha_1\ \mathsf{recurse}\ \alpha_2 \rrbracket_{T,g} & = & \text{union of all sets } \{w\} \times g_k(X) \text{, for } w \in N, \\ & & \text{where } g_k \text{ is obtained in the following manner:} \\ & & g_1 := g[X \mapsto \{v \in N \mid (w,v) \in \llbracket \alpha_1 \rrbracket_{T,g}\}], \\ & & g_{i+1} := g_i[X \mapsto g_i(X) \cup \{v \in N \mid (w,v) \in \llbracket \alpha_2 \rrbracket_{T,g_i}\}] \text{, for } i \geq 1, \\ & & \text{and } k \text{ is the least natural number for which } g_{k+1}{=}g_k. \\ \{\!\!\{ \mathsf{false} \}\!\!\}_{T,g} & = & \emptyset \\ \{\!\!\{ \langle\alpha\rangle \}\!\!\}_{T,g} & = & \{u \in N \mid (u,v) \in \llbracket \alpha \rrbracket_{T,g} \text{ for some } v \in N\} \\ \{\!\!\{ \neg\varphi \}\!\!\}_{T,g} & = & N \setminus \{\!\!\{ \varphi \}\!\!\}_{T,g} \\ \{\!\!\{ \varphi_1 \wedge \varphi_2 \}\!\!\}_{T,g} & = & \{\!\!\{ \varphi_1 \}\!\!\}_{T,g} \cap \{\!\!\{ \varphi_2 \}\!\!\}_{T,g} \\ \{\!\!\{ \varphi_1 \vee \varphi_2 \}\!\!\}_{T,g} & = & \{\!\!\{ \varphi_1 \}\!\!\}_{T,g} \cup \{\!\!\{ \varphi_2 \}\!\!\}_{T,g} \\ \{\!\!\{ X \}\!\!\}_{T,g} & = & g(X), X \in VAR \end{array}$$

$\lhd$

While the semantics $[\![\alpha]\!]_{T,g}$ of a path expression α is defined as a binary relation, it is natural to think of it as a function mapping each node u to a set of nodes $\{v \mid (u,v) \in [\![\alpha]\!]_{T,g}\}$, which we denote by $Result_u^g(\alpha)$. It represents the result of evaluating α in the context node u (using the assignment g). The semantics of the variables and of the WITH operator is most naturally understood from this perspective, and can be equivalently stated as follows:

> $Result_u^g(X) = g(X)$, i.e., when X is used as a path expression, it evaluates to $g(X)$ regardless of the context node.
>
> $Result_u^g(\text{with } X \text{ in } \alpha_1 \text{ recurse } \alpha_2) = X_k$, where $X_1 = Result_u^{g[X\mapsto\emptyset]}(\alpha_1)$, $X_{i+1} = X_i \cup Result_u^{g[X\mapsto X_i]}(\alpha_2)$ for $i \geq 1$, and k is the smallest number such that $X_k = X_{k+1}$.

Note that, at each iteration, the context node of the evaluation of α_1 or α_2 remains u.

When a variable X is used as a node expression, it simply tests whether the current node belongs to the set assigned to X.

The example query below yields the set of nodes that can be reached from the context node by following the transitive closure of the child::a relation.

$$\text{with } X \text{ in child::}a \text{ recurse } X/\text{child::}a$$

The query below yields the set of nodes that are labeled with a and are at an even distance from the context node.

$$(\text{with } X \text{ in . recurse } X/\text{child::*/child::*})/\text{self::}a$$

It is important to note that (unlike MIC) the language provides no way to test whether a given node belongs to the result of with X in α_1 recurse α_2, it only allows to *go to* a node belonging to the result set. From the point of view of XQuery and XPath, it is very natural to define the inflationary fixed point operator in this way, i.e., as an operator on path expressions. However, it has some subtle consequences.

We remark that semantics of the WITH operator we give here differs slighly from the original semantics used in [1]. According to the original semantics, when $Result_u^g(\text{with } \alpha_1 \text{ in } \alpha_2 \text{ recurse })$ is computed, the result of α_1 is only used as a seed of the recursion but is not itself added to the fixed point set. In other words, $Result_u^g(\text{with } X \text{ in } \alpha_1 \text{ recurse } \alpha_2)$ was defined there as X_k, where $X_0 = Result_u^{g[X\mapsto\emptyset]}(\alpha_1)$, $X_1 = Result_u^{g[X\mapsto X_0]}(\alpha_2)$, $X_{i+1} = X_i \cup Result_u^{g[X\mapsto X_i]}(\alpha_2)$ for $i \geq 1$, and k is the least number such that $X_k = X_{k+1}$. The semantics we use here is arguably mathematically more clean and intuitive since it is truly inflationary: all the nodes assigned to the recursion variable during fixed-point computation end up in the result.

2.2 Propositional Modal Logic Extended with IFP (ML+IFP)

The language ML+IFP we consider is an extension of Propositional Modal Logic (ML) [2] with a monadic IFP operator. It is also known as 1MIC, the fragment of Modal Iteration Calculus (MIC) that does not involve simultaneous induction, and it was first introduced in [5], where it was also shown that its satisfiability problem is undecidable on arbitrary structures.

Definition 2.2. ML+IFP Let Σ be a set of labels and VAR a set of variables. Then the syntax of ML+IFP is defined as follows:

$$\varphi \ ::= \ \bot \mid l \mid X \mid \Diamond\varphi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid (\text{ifp}\, X \leftarrow \varphi)$$

where $l \in \Sigma$, $X \in VAR$.

The semantics of ML+IFP is given in terms of Kripke models. To facilitate the comparison with CXP+IFP, we will assume that the Kripke models assign a unique label to each node, rather than a set of labels. This is not essential. Let $T = (N,R,L)$ be a Kripke model, where N is a set of nodes, $R \subseteq N \times N$ is a binary relation on the nodes in N, and L is a valuation function that assigns a label from Σ to to each in N. Let $g(\cdot)$ be an assignment function from variables to sets of nodes, $g : VAR \longrightarrow \wp(N)$. Then the semantics of ML+IFP formulas are as follows:

$$\begin{array}{rcl}
[\![\bot]\!]_{T,g} & = & \emptyset \\
[\![l]\!]_{T,g} & = & \{n \in N \mid L(n) = l\} \\
[\![X]\!]_{T,g} & = & g(X) \\
[\![\Diamond\varphi]\!]_{T,g} & = & \{u \mid \exists v.(u,v) \in R \wedge v \in [\![\varphi]\!]_{T,g}\} \\
[\![\neg\varphi]\!]_{T,g} & = & N \setminus [\![\varphi]\!]_{T,g} \\
[\![\varphi_1 \wedge \varphi_2]\!]_{T,g} & = & [\![\varphi_1]\!]_{T,g} \cap [\![\varphi_2]\!]_{T,g} \\
[\![\mathsf{ifp}\ X \leftarrow \varphi]\!]_{T,g} & = & g_k(X), \text{ where } g_k \text{ is obtained in the following manner:} \\
 & & g_0 := g[X \mapsto \emptyset], \\
 & & g_{i+1} := g_i[X \mapsto g_i(X) \cup [\![\varphi]\!]_{T,g_i}], \text{ for } i \geq 0, \\
 & & \text{where } k \text{ is the minimum number for which } g_{k+1}{=}g_k.
\end{array}$$

We write $T,g,u \Vdash \varphi$ if $v \in [\![\varphi]\!]_{T,g}$. If a formula has no free variables, we may leave out the assignment and write $T,u \Vdash \varphi$ or $u \in [\![\varphi]\!]_T$. ◁

It was shown in [5] that the satisfiability problem for ML+IFP on arbitrary Kripke models is highly undecidable. As we will show below, it is undecidable on finite trees as well.

3 Relationship between ML+IFP and CXP+IFP

In this section, we give a truth-preserving translation from ML+IFP to CXP+IFP. In fact, the translation yields CXP+IFP expressions that use only the *self* and *descendant* axes. It follows that this fragment of CXP+IFP has already (at least) the expressive power of ML+IFP.

One of the main differences between ML+IFP and CXP+IFP is that, in the former, fixed-point expressions are node expressions that test whether the current node belongs to the fixed point of a formula, while in the latter, fixed-point expressions are path expressions that travel to nodes belonging to the fixed point of a formula. Another difference is that, in CXP+IFP, during the entire fixed point computation, the expressions are evaluated from a fixed context node, whereas in ML+IFP, whether a node is added to the set at some stage of the fixed point computation is determined by local properties of the subtree below that node.

In our translation from ML+IFP to CXP+IFP we have to overcome these differences. The main idea for the translation of ML+IFP formulas of the form $\mathsf{ifp}\ X \leftarrow \varphi$ will be that, during the fixed point computation, we treat leaf nodes in a special way, never adding them to the fixed point set but keeping track of them separely. More precisely, we first compute the set Y of all leaf nodes satisfying $\mathsf{ifp}\ X \leftarrow \varphi$. Next, we let $X_0 = \emptyset$ and X_{i+1} is computed as $X_i \cup ([\![\varphi]\!]_{T,g[X \mapsto X_i \cup Y]} - Y)$. Observe how the nodes in Y are added to the input and substracted again from the output. Let X_k be the fixed point of the sequence $X_0 \subseteq X_1 \subseteq \cdots$. Then we have that $[\![\mathsf{ifp}\ X \leftarrow \varphi]\!]_{T,g} = X_k \cup Y$. The advantage of this construction is that, since the leafs are never added during the fixed point computation, they can be freely used for signalling that the context node was added to the set X: if the context node is added at some stage, we add a leaf node as well, and the presence of a leaf node in the result set will be used as a sign that we test for afterwards.

Before we give the details of the construction, we first note that when computing the inflationary fixed point of an ML+IFP formula, any leaf node that is added to the fixed point set is in fact already added at the first stage of the fixed point computation. This is expressed by the following lemma.

Lemma 3.1. *Let u be any node in a Kripke model T, and let $\varphi(X)$ be any ML+IFP formula and g an assignment. If u has no successors, then $u \in [\![\textsf{ifp}\, X \leftarrow \varphi]\!]_{T,g}$ iff $u \in [\![\varphi]\!]_{T,g[X\mapsto\emptyset]}$.*

In what follows we will use $\oslash$ as shorthand for $\textsf{self::}*[\textsf{false}]$, $\textsf{desc-or-self::}*$ as shorthand for $\textsf{desc::}*\cup\textsf{self::}*$, and $\textsf{leaf}$ as shorthand for $\neg\langle\textsf{child::}*\rangle$. Also, for node expressions φ, ψ and a variable X, such that X only occurs in φ in the form of node tests, we will denote by $\varphi^{X/\psi}$ the node expression obtained from by replacing all free occurrences of X in φ by the node expression ψ.

The translation $\tau(\cdot)$ from ML+IFP formulas to CXP+IFP node expressions is given by Equation (1).

$$\begin{array}{rcl}
\tau(\bot) & = & \textsf{false} \\
\tau(l) & = & \langle\textsf{self::}l\rangle \\
\tau(\varphi_1\wedge\varphi_2) & = & \tau(\varphi_1)\wedge\tau(\varphi_2) \\
\tau(\neg\varphi) & = & \neg\tau(\varphi) \\
\tau(X) & = & X \\
\tau(\Diamond\varphi) & = & \langle\textsf{child::}*[\tau(\varphi)]\rangle \\
\tau\big(\textsf{ifp}\, X\leftarrow\varphi\big) & = & \langle\big(\textsf{with}\; X\; \textsf{in desc-or-self::*}[\tau(\varphi)^{X/\textsf{false}}\wedge\neg\textsf{leaf}]\;\textsf{recurse} \\
& & \qquad \textsf{desc-or-self::*}[\tau(\varphi)^{X/(X\vee\tau(\varphi)_{\textsf{leaf}})}\wedge\neg\textsf{leaf}]\cup \\
& & \qquad \textsf{self::}*[X\vee\tau(\varphi)_{\textsf{leaf}}]/\textsf{desc::}* \qquad\qquad\qquad \big)[\textsf{leaf}]\rangle \\
& & \text{where } \tau(\varphi)_{\textsf{leaf}} = \tau(\varphi)^{X/\textsf{false}}\wedge\textsf{leaf}
\end{array} \tag{1}$$

Theorem 3.2. *Let $T=(N,R,L)$ be a node-labeled finite tree, g an assignment, and u a node in T. Then $T,g,u \Vdash \varphi \iff T,g,u \Vdash \tau(\varphi)$.*

We can conclude that CXP+IFP node expressions have (at least) the expressive power of ML+IFP. Since the desc axis is definable from the child axis, the same holds of course for the fragment of CXP+IFP without the desc axis. What is more surprising is that the same holds for the fragment of CXP+IFP without the child axis. The next Lemma shows that the use of the child axis in the above translation can be avoided (provided that we keep, of course, the desc axis). Note that the child axis was only used in the translation of formulas of the form $\Diamond\varphi$.

Proposition 3.3. *For any node expression φ, $\langle\textsf{child::}*[\varphi]\rangle$ is equivalent to the following node expression (which does not use the child axis):*

$$\begin{array}{c}
\langle\Big(\textsf{with}\, X\; \textsf{in desc::}*/\textsf{desc::}*[\textsf{leaf}]\;\textsf{recurse self::}*[\langle\textsf{desc::}*[\textsf{leaf}\wedge\neg X\wedge\varphi]\rangle]\Big)[\neg\textsf{leaf}]\rangle \\
\vee \\
\langle\Big(\textsf{with}\, X\; \textsf{in desc::}*/\textsf{desc::}*[\neg\textsf{leaf}]\;\textsf{recurse desc::}*[\neg\textsf{leaf}\wedge\neg X\wedge\varphi]/\textsf{desc::}*\Big)[\textsf{leaf}]\rangle
\end{array}$$

4 The Undecidability of CXP+IFP and of ML+IFP on Finite Trees

We show that the satisfiability problem for ML+IFP on finite trees is undecidable, and therefore also (by our earlier translation), the satisfiability problem for CXP+IFP.

Theorem 4.1. *The satisfiability problem of ML+IFP on finite trees is undecidable.*

Corollary 4.2. *The satisfiability problem of CXP+IFP is undecidable, even if the child axis is disllowed.*

The proof is based on a reduction from the halting problem for 2-register machines (cf. [4]). A 2-register machine is a very simple kind of deterministic automaton without input and output. It has two registers containing integer values, and instructions for incrementing and decrementing the content of the registers. These 2-register automata form one of the simplest types of machines for which the halting problem is already undecidable. The formal definition is as follows:

A *2-register machine* M is a tuple $M = (Q, \delta, q_0, q_f)$, where Q is a finite set of *states*, δ is a *transition function* from Q to a set of instructions I, defined below, and q_0, q_f are designated states in Q, called *initial and final states*, respectively. The set of *instructions* I consists of four kinds of instructions:

$INC_A(q')$: increment the value stored in A and move to state q';

$INC_B(q')$: increment the value stored in B and move to state q';

$DEC_A(q', q'')$: if the value stored in A is bigger than 0 then decrement it by one and move to state q', otherwise move to state q'' without changing the value in A nor B; and

$DEC_B(q', q'')$: if the value stored in B is bigger than 0 then decrement it by one and move to state q', otherwise move to state q'' without changing the value in A nor B.

The problem whether a given two-register machine M has a successful run (starting in the initial state with both register values 0, and ending in the final state with both register values 0) is undecidable.

A run of M can be represented as a string over the alphabet $Q \cup \{a, b, \$\}$ of the form $q_1\vec{a}_1\vec{b}_1 \dots q_n\vec{a}_n\vec{b}_n\$$, where each $q_i \in Q$ represents the state of the automaton at the i-th step, and $\vec{a}_i, \vec{b}_i$ are sequences of as respectively bs whose length represents the register content at the i-th step ($ is used to mark the end of the string). We construct an ML+IFP formula which expresses that *for each branch from the current node, the string consisting of the letters of the nodes on the branch encodes an accepting run of the 2-register machine*. It follows that the formula is satisfiable if and only if M has a successful run.

5 Discussion

One natural follow-up question is whether CXP+IFP node expressions are strictly more expressive than ML+IFP formulas.

Other natural follow-up questions concern fragments of CXP+IFP. Recall that in CXP+IFP, the variables can be used both as atomic path expressions and as atomic node expressions. The former is the most natural, but translation we gave from ML+IFP to CXP+IFP crucially uses the latter. We are currently investigating the fragment of CXP+IFP in which variables are only allowed as atomic path expressions.

It is also natural to consider CXP+IFP expressions where the fixed point variables occur only under an even number of negations, so that the WITH-operator computes the least fixed point of a monotone operation. Note that this fragment is decidable, since it is contained in monadic second-order logic.

Acknowledgements

The first author is supported by the Netherlands Organization of Scientific Research (NWO) grant 017.001.190. The second author is supported NWO grant 639.021.508 and by ERC Advanced Grant Webdam on Foundation of Web data management. We thank Anuj Dawar for a useful discussion.

References

[1] L. Afanasiev, T. Grust, M. Marx, J. Rittinger, and J. Teubner. Recursion in XQuery: put your distributivity safety belt on. In M. L. Kersten, et al., eds., *Proc. of 12th Int. Conf. on Extending Database Technology, EDBT 2009 (St. Petersburg, March 2009)*, v. 360 of *ACM Int. Conf. Proc. Series*, pp. 345–356. ACM Press, 2009.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, v. 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 2001.

[3] P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teuber. MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In *Proc. of ACM SIGMOD Int. Conf. on Management of Data (Chicago, IL, June 2006)*, pp. 479–490. ACM Press, 2006.

[4] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[5] A. Dawar, E. Grädel, and S. Kreutzer. Inflationary fixed points in modal logic. *ACM Trans. on Comput. Logic*, 5(2):282–315, 2004.

[6] G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *Proc. of 17th Ann. IEEE Symp. on Logic in Computer Science, LICS 2002 (Copenhagen, July 2002)*, pp. 189–202. IEEE CS Press, 2002.

Solutions of Generalized Recursive Metric-Space Equations*

Lars Birkedal, Kristian Støvring, and Jacob Thamsborg
IT University of Copenhagen,
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
birkedal@itu.dk, kss@itu.dk, and thamsborg@itu.dk

Abstract

It is well known that one can use an adaptation of the inverse-limit construction to solve recursive equations in the category of complete ultrametric spaces. We show that this construction generalizes to a large class of categories with metric-space structure on each set of morphisms: the exact nature of the objects is less important. In particular, the construction immediately applies to categories where the objects are ultrametric spaces with 'extra structure', and where the morphisms preserve this extra structure. The generalization is inspired by classical domain-theoretic work by Smyth and Plotkin. Our primary motivation for solving generalized recursive metric-space equations comes from recent and ongoing work on Kripke-style models in which the sets of worlds must be recursively defined.

For many of the categories we consider, there is a natural subcategory in which each set of morphisms is required to be a compact metric space. Our setting allows for a proof that such a subcategory always inherits solutions of recursive equations from the full category.

As another application, we present a construction that relates solutions of generalized domain equations in the sense of Smyth and Plotkin to solutions of equations in our class of categories.

1 Introduction

Smyth and Plotkin [17] showed that in the classical inverse-limit construction of solutions to recursive domain equations, what matters is not that the *objects* of the category under consideration are domains, but that the sets of *morphisms* between objects are domains. In this work we show that, in the case of ultrametric spaces, the standard construction of solutions to recursive metric-space equations [5, 10] can be similarly generalized to a large class of categories with metric-space structure on each set of morphisms.

The generalization in particular allows one to solve recursive equations in categories where the objects are ultrametric spaces with some form of additional structure, and where the morphisms preserve this additional structure. Our main motivation for solving equations in such categories comes from recent and ongoing work in denotational semantics by the authors and others [7, 15]. There, solutions to such equations are used in order to construct Kripke models over recursively defined worlds: a novel approach that allows one to give semantic models of predicates and relations over languages with dynamically allocated, higher-order store. See [8] for examples of such applications.

For many of the categories we consider, there is a natural variant, indeed a subcategory, in which each set of morphisms is required to be a compact metric space [2, 9]. Our setting allows for a general proof that such a subcategory inherits solutions of recursive equations from the full category. Otherwise put, the problem of solving recursive equations in such a 'locally compact' subcategory is, in a certain sense, reduced to the similar problem for the full category. The fact that one can solve recursive equations in a category of compact ultrametric spaces [9] arises as a particular instance. (For various applications of compact metric spaces in semantics, see the references in the introduction to van Breugel and Warmerdam [9].)

*See the full article for proofs and further details [8].

As another application, we present a construction that relates solutions of generalized domain equations in the sense of Smyth and Plotkin to solutions of equations in our class of categories. This construction generalizes and improves an earlier one due to Baier and Majster-Cederbaum [6].

The key to achieving the right level of generality in the results lies in inspiration from enriched category theory. We shall not refer to general enriched category theory below, but rather present the necessary definitions in terms of metric spaces. The basic idea is, however, that given a cartesian category V (or more generally, a monoidal category), one considers so-called V-categories, in which the 'hom-sets' are in fact objects of V instead of sets, and where the 'composition functions' are morphisms in V.

Other related work. The idea of considering categories with metric spaces as hom-sets has been used in earlier work [14, 9]. Rutten and Turi [14] show existence and uniqueness of fixed points in a particular category of (not necessarily ultrametric) metric spaces, but with a proof where parts are more general. In other work, van Breugel and Warmerdam [9] show uniqueness for a more general notion of categories than ours, again not requiring ultrametricity. Neither of these articles contain a theorem about existence of fixed points for a general class of 'metric-enriched' categories (as in our Theorem 3.1), nor a general theorem about fixed points in locally compact subcategories (Theorem 4.1.)

Alessi et al. [3] consider solutions to *non-functorial* recursive equations in certain categories of metric spaces, i.e., recursive equations whose solutions cannot necessarily be described as fixed-points of functors. In contrast, we only consider *functorial* recursive equations in this work.

Wagner [18] gives a comprehensive account of a generalized inverse limit construction that in particular works for categories of metric spaces and categories of domains. Another such construction has recently been given by Kostanek and Waszkiewicz [11]. Our generalization is in a different direction, namely to categories where the hom-sets are metric spaces. We do not know whether there is a common generalization of our work and Wagner's work; in this work we do not aim for maximal generality, but rather for a level of generality that seems right for our applications [8].

2 Ultrametric Spaces

We first recall some basic definitions and properties about metric spaces [13, 16]. A metric space (X,d) is 1-*bounded* if $d(x,y) \leq 1$ for all x and y in X. We shall only work with 1-bounded metric spaces. One advantage of doing so is that one can define coproducts and general products of such spaces; alternatively, one could have allowed infinite distances.

An *ultrametric space* is a metric space (X,d) that satisfies the 'ultrametric inequality' $d(x,z) \leq \max(d(x,y),d(y,z))$ and not just the weaker triangle inequality (where one has $+$ instead of max on the right-hand side). It might be helpful to think of the function d of an ultrametric space (X,d) not as a measure of (euclidean) distance between elements, but rather as a measure of the degree of similarity between elements.

Let CBUlt be the category with complete, 1-bounded ultrametric spaces as objects and non-expansive (i.e., non-distance-increasing) functions as morphisms [5]. This category is cartesian closed [16]; here one needs the ultrametric inequality. The terminal object is the one-point space. Binary products are defined in the natural way: the distance between two pairs of elements is the maximum of the two pointwise distances. The exponential $A \to B$, sometimes written B^A, has the set of non-expansive functions from A to B as the underlying set, and the 'sup'-metric $d_{A\to B}$ as distance function: $d_{A\to B}(f,g) = \sup\{d_B(f(x),g(x)) \mid x \in A\}$. For both products and exponentials, limits are pointwise. It follows from the cartesian closed structure that the function $C^B \times B^A \to C^A$ given by composition is non-expansive; this fact is needed in several places below.

2.1 M-Categories

The basic idea of this work is to generalize a theorem about a particular category of metric spaces to a theorem about more general categories where each hom-set is an ultrametric space. In analogy with the O-categories of Smyth and Plotkin (O for 'order' or 'ordered') we call such categories M-categories.

Definition 2.1. An *M-category* is a category $\mathscr{C}$ where each hom-set $\mathscr{C}(A,B)$ is equipped with a distance function turning it into a non-empty, complete, 1-bounded ultrametric space, and where each composition function $\circ : \mathscr{C}(B,C) \times \mathscr{C}(A,B) \to \mathscr{C}(A,C)$ is non-expansive with respect to these metrics. (Here the domain of such a composition function is given the product metric.)

Notice that the hom-sets of an M-category are required to be *non-empty* metric spaces. This restriction allows us to avoid tedious special cases in the results below since the proofs depend on Banach's fixed-point theorem.

The simplest example of an M-category is the category $\mathsf{CBUlt}_{\mathrm{ne}}$ of non-empty, 1-bounded, complete ultrametric spaces and non-expansive maps. Here the distance function on each hom-set $\mathsf{CBUlt}_{\mathrm{ne}}(A,B)$ is given by $d(f,g) = \sup\{d_B(f(x),g(x)) \mid x \in A\}$. The category $\mathsf{CBUlt}_{\mathrm{ne}}$ is cartesian closed since CBUlt is: it suffices to verify that CBUlt-products of non-empty metric spaces are non-empty, and similarly for exponentials.

Let $\mathscr{C}$ be an M-category. A functor $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ is *locally contractive* if there exists some $c < 1$ such that $d(F(f,g),F(f',g')) \leq c \cdot \max(d(f,f'),d(g,g'))$ for all f,f', g, and g'. Notice that the same c must work for all hom-sets of $\mathscr{C}$.

3 Solving Recursive Equations

Let $\mathscr{C}$ be an M-category. We consider mixed-variance functors $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ on $\mathscr{C}$ and recursive equations of the form $X \cong F(X,X)$. In other words, given such an F we seek a fixed point of F up to isomorphism.

Covariant endofunctors on $\mathscr{C}$ are a special case of mixed-variance functors. It would in some sense suffice to study covariant functors: if $\mathscr{C}$ is an M-category, then so are $\mathscr{C}^{\mathrm{op}}$ (with the same metric on each hom-set as in $\mathscr{C}$) and $\mathscr{C}^{\mathrm{op}} \times \mathscr{C}$ (with the product metric on each hom-set), and it is well-known how to construct a 'symmetric' endofunctor on $\mathscr{C}^{\mathrm{op}} \times \mathscr{C}$ from a functor such as F above. We explicitly study mixed-variance functors since the proof of the existence theorem below would in any case involve an M-category of the form $\mathscr{C}^{\mathrm{op}} \times \mathscr{C}$. As a benefit we directly obtain theorems of the form useful in applications. For example, for the existence theorem we are interested in completeness conditions on $\mathscr{C}$, not on $\mathscr{C}^{\mathrm{op}} \times \mathscr{C}$.

3.1 Uniqueness of Solutions

Our results below depend on the assumption that the given functor F on $\mathscr{C}$ is locally contractive. One easy consequence of this assumption is that, unlike in the domain-theoretic setting [17], there is at most one fixed point of F up to isomorphism.

Theorem 3.1. *Let $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ be a locally contractive functor on an M-category $\mathscr{C}$, and assume that $i : F(A,A) \to A$ is an isomorphism. Then the pair (i,i^{-1}) is a* bifree algebra *for F in the following sense: for all objects B of $\mathscr{C}$ and all morphisms $f : F(B,B) \to B$ and $g : B \to F(B,B)$, there exists a* *unique pair of morphisms* $(k : B \to A, h : A \to B)$ *such that* $h \circ i = f \circ F(k,h)$ *and* $i^{-1} \circ k = F(h,k) \circ g$*:*

$$\begin{tikzcd}
F(A,A) \arrow[r, shift left, "F(k,h)"] \arrow[d, shift left, "i"] & F(B,B) \arrow[l, shift left, "F(h,k)"] \arrow[d, shift left, "f"] \\
A \arrow[u, shift left, "i^{-1}"] \arrow[r, dashed, shift left, "h"] & B \arrow[l, dashed, shift left, "k"] \arrow[u, shift left, "g"]
\end{tikzcd}$$

In particular, A is the unique fixed point of F up to isomorphism.

3.2 Existence of Solutions

In the existence theorem for fixed points of contractive functors, the M-category $\mathscr{C}$ will be assumed to satisfy a certain completeness condition involving limits of ω^{op}-chains. Since there are different M-categories satisfying more or less general variants of this condition, it is convenient to present the existence theorem in a form that lists a number of successively weaker conditions.

An *increasing Cauchy tower* is a diagram

$$\begin{tikzcd}
A_0 \arrow[r, shift left, "f_0"] & A_1 \arrow[l, shift left, "g_0"] \arrow[r, shift left, "f_1"] & \cdots \arrow[l, shift left, "g_1"] \arrow[r, shift left, "f_{n-1}"] & A_n \arrow[l, shift left, "g_{n-1}"] \arrow[r, shift left, "f_n"] & \cdots \arrow[l, shift left, "g_n"]
\end{tikzcd}$$

where $g_n \circ f_n = id_{A_n}$ for all n, and where $\lim_{n\to\infty} d(f_n \circ g_n,\, id_{A_{n+1}}) = 0$. Notice that this definition only makes sense for M-categories. The M-category $\mathscr{C}$ has *inverse limits of increasing Cauchy towers* if for every such diagram, the sub-diagram containing only the arrows g_n has a limit. (This subdiagram is, incidentally, an ω^{op}-chain of morphisms that are split epi, i.e., have a left inverse.)

Theorem 3.2. *Assume that the M-category* $\mathscr{C}$ *satisfies any of the following (successively weaker) conditions:*

1. $\mathscr{C}$ *is complete.*
2. $\mathscr{C}$ *has a terminal object and limits of* ω^{op}*-chains.*
3. $\mathscr{C}$ *has a terminal object and limits of* ω^{op}*-chains of split epis.*
4. $\mathscr{C}$ *has a terminal object and inverse limits of increasing Cauchy towers.*

Then every locally contractive functor $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ *on* $\mathscr{C}$ *has a unique fixed point up to isomorphism.*

4 Locally Compact Subcategories of M-Categories

The condition in Theorem 3.2 that involves Cauchy towers is included in order to accommodate categories where the hom-sets are compact ultrametric spaces [2, 9]: one example is the full subcategory $\mathsf{KBUlt}_{\mathrm{ne}}$ of *compact*, non-empty, 1-bounded ultrametric spaces. This subcategory is merely the simplest example of a full, 'locally compact' subcategory of an M-category. Such a subcategory always inherits fixed points of functors from the full category:

Theorem 4.1. *Assume that* $\mathscr{C}$ *is an M-category with a terminal object and limits of* ω^{op}*-chains of split epis. Let I be an arbitrary object of* $\mathscr{C}$*, and let* $\mathscr{D}$ *be the full subcategory of* $\mathscr{C}$ *consisting of those objects A such that the metric space* $\mathscr{C}(I,A)$ *is compact.* $\mathscr{D}$ *is an M-category with limits of increasing Cauchy towers, and hence every locally contractive functor* $F : \mathscr{D}^{\mathrm{op}} \times \mathscr{D} \to \mathscr{D}$ *has a unique fixed point up to isomorphism.*

For a monoidal closed $\mathscr{C}$, the tensor unit is an appropriate choice of I. In particular, taking $\mathscr{C}$ to be $\mathsf{CBUlt}_{\mathrm{ne}}$ and I to be one-point metric space, one obtains:

Corollary 4.2 ([9]). *Every locally contractive functor $F : \mathsf{KBUlt}_{\mathrm{ne}}{}^{\mathrm{op}} \times \mathsf{KBUlt}_{\mathrm{ne}} \to \mathsf{KBUlt}_{\mathrm{ne}}$ has a unique fixed point up to isomorphism.*

5 Domain Equations: from O-Categories to M-Categories

As another illustration of M-categories, we present a general construction that gives for every O-category $\mathscr{C}$ (see below) a derived M-category $\mathscr{D}$. In addition, the construction gives for every locally continuous mixed-variance functor F on $\mathscr{C}$ a locally contractive mixed-variance functor G on $\mathscr{D}$ such that a fixed point of G (necessarily unique, by Theorem 3.1) is the same as a fixed point of F that furthermore satisfies a 'minimal invariance' condition [12]. Thus, generalized domain equations can be solved in M-categories.

The construction generalizes an earlier one [6] which is for the particular category of pointed cpos and strict, continuous functions (or full subcategories thereof) and only works for a restricted class of functors that does not include general function spaces.

Rank-ordered cpos [6], independently discovered under the name 'uniform cpos' [7], arise from a particular instance of an M-category obtained from this construction. The extra metric information in that category (as compared with the underlying O-category) is useful in realizability models [4, 1].

An *O-category* [17] is a category $\mathscr{C}$ where each hom-set $\mathscr{C}(A,B)$ is equipped with an ω-complete partial order, usually written $\sqsubseteq$, and where each composition function is continuous with respect to these orders. A functor $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ is *locally continuous* if each function on hom-sets that it induces is continuous.

Assume now that $\mathscr{C}$ is an O-category such that each hom-set $\mathscr{C}(A,B)$ contains a least element $\bot_{A,B}$ and such that the composition functions of $\mathscr{C}$ are strict: $f \circ \bot_{A,B} = \bot_{A,C} = \bot_{B,C} \circ g$ for all f and g. We construct an M-category $\mathscr{D}$ of 'rank-ordered $\mathscr{C}$-objects' as follows. An object $(A,(\pi_n)_{n\in\omega})$ of $\mathscr{D}$ is a pair consisting of an object A of $\mathscr{C}$ and a family of endomorphisms $\pi_n : A \to A$ in $\mathscr{C}$ that satisfies the following requirements:

(1) $\pi_0 = \bot_{A,A}$.

(2) $\pi_m \sqsubseteq \pi_n$ for all $m \leq n$.

(3) $\pi_m \circ \pi_n = \pi_n \circ \pi_m = \pi_{\min(m,n)}$ for all m and n.

(4) $\bigsqcup_{n\in\omega} \pi_n = id_A$.

Then, a morphism from $(A,(\pi_n)_{n\in\omega})$ to $(A',(\pi'_n)_{n\in\omega})$ in $\mathscr{D}$ is a morphism f from A to A' in $\mathscr{C}$ satisfying that $\pi'_n \circ f = f \circ \pi_n$ for all n. Composition and identities in $\mathscr{D}$ are the same as in $\mathscr{C}$. Finally, the distance function on a hom-set $\mathscr{D}((A,(\pi_n)_{n\in\omega}),\,(A',(\pi'_n)_{n\in\omega}))$ is defined as follows: $d(f,g) = 2^{-\max\{n\in\omega \mid \pi'_n\circ f = \pi'_n\circ g\}}$ if $f \neq g$, and $d(f,g) = 0$ otherwise. (One can show using conditions (1)-(4) above that this function is in fact well-defined.)

Proposition 5.1. *$\mathscr{D}$ is an M-category.*

Now let $F : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{C}$ be a locally continuous functor. We construct a locally contractive functor $G : \mathscr{D}^{\mathrm{op}} \times \mathscr{D} \to \mathscr{D}$ from F. On objects, G is given by

$$G((A,(\pi^A_n)_{n\in\omega}),\,(B,(\pi^B_n)_{n\in\omega})) = (F(A,B),(\pi^{A,B}_n)_{n\in\omega})$$

where $\pi^{A,B}_0 = \bot$ and $\pi^{A,B}_{n+1} = F(\pi^A_n,\pi^B_n)$ for all n. On morphisms, G is the same as F, i.e., $G(f,g) = F(f,g)$. One can verify that G is well-defined and furthermore locally contractive with factor $1/2$.

Proposition 5.2. *Let A be an object of $\mathscr{C}$. The following two conditions are equivalent. (1) There exists an isomorphism $i : F(A,A) \to A$ such that $id_A = \mathit{fix}(\lambda e^{\mathscr{C}(A,A)}.\, i \circ F(e,e) \circ i^{-1})$. (Here fix is the least-fixed-point operator.) (2) There exists a family of morphisms $(\pi_n)_{n\in\omega}$ such that $\overline{A} = (A, (\pi_n)_{n\in\omega})$ is the unique fixed-point of G up to isomorphism.*

It remains to discuss how completeness properties of $\mathscr{C}$ transfer to $\mathscr{D}$. One can show that the forgetful functor from $\mathscr{D}$ to $\mathscr{C}$ creates terminal objects and limits of ω^{op}-chains of split epis. Alternatively, by imposing an additional requirement on $\mathscr{C}$ one can show that the forgetful functor creates *all* limits: for a given limit in $\mathscr{C}$, the induced bijection between cones and mediating morphisms must be an isomorphism in the category of cpos (where cones are ordered pointwise, using the order on each hom-set). That requirement is in particular satisfied by the usual concrete categories of cpos.

References

[1] M. Abadi and G. D. Plotkin. A per model of polymorphism and recursive types. In *Proc. of 5th Annual IEEE Symp. on Logic in Computer Science, LICS '90 (Philadelphia, PA, June 1990)*, pp. 355–365. IEEE CS Press, 1990.

[2] F. Alessi, P. Baldan, and G. Bellè. A fixed-point theorem in a category of compact metric spaces. *Theor. Comput. Sci.*, 146(1–2):311–320, 1995.

[3] F. Alessi, P. Baldan, G. Bellè, and J. J. M. M. Rutten. Solutions of functorial and non-functorial metric domain equations. *Electron. Notes in Theor. Comput. Sci.*, 1:1–12, 1995.

[4] R. M. Amadio. Recursion over realizability structures. *Inform. and Comput.*, 91(1):55–85, 1991.

[5] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. of Comput. and Syst. Sci.*, 39(3):343–375, 1989.

[6] C. Baier and M. E. Majster-Cederbaum. The connection between initial and unique solutions of domain equations in the partial order and metric approach. *Formal Aspects of Computing*, 9(4):425–445, 1997.

[7] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In L. de Alfaro, ed., *Proc. of 12th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2009 (York, March 2009)*, v. 5504 of *Lect. Notes in Comput. Sci.*, pp. 456–470. Springer, 2009.

[8] L. Birkedal, K. Støvring, and J. Thamsborg. The category-theoretic solution of recursive metric-space equations, 2009. Manuscript, submitted to journal. Available at `http://itu.dk/people/kss/papers/metric-equations.pdf`.

[9] F. van Breugel and J. Warmerdam. Solving domain equations in a category of compact metric spaces. Report CS-R9424, CWI, Amsterdam, 1994.

[10] J. W. de Bakker and J. Zucker. Processes and the denotational semantics of concurrency. *Inform. and Control*, 54:70–120, 1982.

[11] M. Kostanek and P. Waszkiewicz. On the influence of domain theory on *Q*-categories, 2009. Manuscript, submitted. Available at `http://tcs.uj.edu.pl/Waszkiewicz/`.

[12] A. M. Pitts. Relational properties of domains. *Inform. and Comput.*, 127(2):66–90, 1996.

[13] J. J. M. M. Rutten. Elements of generalized ultrametric domain theory. *Theor. Comput. Sci.*, 170(1–2):349–381, 1996.

[14] J. J. M. M. Rutten and D. Turi. On the foundations of final semantics: non-standard sets, metric spaces, partial orders. Report CS-R9241, CWI, Amsterdam, 1992.

[15] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *Proc. of 23rd Int. Wksh. on Computer Science Logic (Coimbra, Sept. 2009)*, *Lect. Notes in Comput. Sci.*, Springer, to appear.

[16] M. B. Smyth. Topology. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, eds., *Handbook of Logic in Computer Science, Vol. 1: Background: Mathematical Structures*, pp. 641–761. Oxford Univ. Press, 1992.

[17] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. on Comput.*, 11(4):761–783, 1982.

[18] K. R. Wagner. *Solving Recursive Domain Equations with Enriched Categories*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1994.

Scattered Algebraic Linear Orderings

S. L. Bloom
Department of Computer Science, Stevens Institute of Technology,
Hoboken, NJ 07030, USA
bloom@cs.stevens.edu
Zoltán Ésik[*]
Department of Informatics, University of Szeged,
P. O. Box 652, H-6701 Szeged, Hungary
ze@inf.u-szeged.hu

Abstract

An algebraic linear ordering is a component of the initial solution of a first-order recursion scheme over the continuous categorical algebra of countable linear orderings equipped with the sum operation and the constant 1. Due to a general Mezei-Wright type result, algebraic linear orderings are exactly those isomorphic to the linear ordering of the leaves of an algebraic tree. Moreover, using a result of Courcelle together with a Mezei-Wright type result, we can show that the algebraic words are exactly those that are isomorphic to the lexicographic ordering of a deterministic context-free language. Algebraic well-orderings have been shown to be those well-orderings whose order type is less than $\omega^{\omega^{\omega}}$. We prove that the Hausdorff rank of any scattered algebraic linear ordering is less than ω^{ω}.

1 Introduction

Fixed points and finite systems of fixed point equations occur in just about all areas of computer science. Regular and context-free languages, rational and algebraic formal power series, finite state process behaviors can all be characterized as (components of) canonical solutions (e.g., unique, least or greatest, or initial or final solutions) of systems of fixed point equations, or recursion schemes.

In this paper we consider systems fixed point equations over countable linear orderings. Consider for example the system

$$\begin{aligned} X &= X+Y+X \\ Y &= \mathbf{1}+Y \end{aligned}$$

where $+$ denotes the usual sum operation on linear orderings, and $\mathbf{1}$ is a singleton linear ordering. It has no solution among finite linear orderings, but it has many solutions among countable linear orderings. The second component of the simplest "canonical" solution is the ordering $\mathbb{N}$ of the nonnegative integers, whereas the first component is the ordering obtained from the ordering $\mathbb{Q}$ of the rationals by replacing each point with a copy of $\mathbb{N}$.

In the above "regular" system of equations, the unknowns X,Y range over linear orderings. More generally, in an "algebraic" or "first-order" scheme we allow unknowns ranging over functions, or rather, functors defined on linear orderings:

$$\begin{aligned} X &= Y(\mathbf{1}) \\ Y(x) &= Z(x)+Y(\mathbf{1}+x) \\ Z(x) &= Z(x)+x+Z(x) \end{aligned}$$

[*] Supported in part by grant no. K 75249 from the National Foundation of Hungary for Scientific Research.

Here, X ranges over linear orderings, while Y,Z range over functions (or more precisely, over functors) on linear orderings. The first component of the canonical solution of this system is the linear ordering $L_1 + L_2 + \ldots$, where for each $n > 0$, L_n is the linear ordering obtained from $\mathbb{Q}$ by replacing each point with the linear ordering $\mathbf{n}$, the n-fold sum of $\mathbf{1}$ with itself.

Regular linear orderings are a special case of the regular words (or arrangements) of Courcelle [9]. Regular words and linear orders were studied in [18, 16, 3, 4]. The study of algebraic words and linear orderings was initiated in [5]. As an application of a general Mezei-Wright type result [6], one obtains that a linear ordering is algebraic (regular) iff it is isomorphic to the linear ordering of the leaves of an algebraic (regular) tree. (See [10, 15] for the definition of algebraic and regular trees.)

In this paper, we first review the characterization of algebraic linear orderings by deterministic context-free languages. Then we show that the Hausdorff rank of every **scattered** algebraic linear ordering is less than ω^ω. This extends one direction of a result of [7] where it is shown that an ordinal is algebraic iff it is less than ω^{ω^ω}. As a consequence of our results, we also obtain that if a scattered linear ordering is isomorphic to the ordering of a deterministic context-free language, then its Hausdorff rank is less than ω^ω.

2 Basic Notions and Notation

2.1 Continuous Categorical Σ-Algebras

Suppose that $\Sigma = \bigcup_{n\geq 0} \Sigma_n$ is a ranked alphabet. A **categorical Σ-algebra** ([4, 5, 6]) $\mathscr{A}$ consists of a (small) category, also denoted $\mathscr{A}$ together with a functor $\sigma^{\mathscr{A}} : \mathscr{A}^n \to \mathscr{A}$, for each letter $\sigma \in \Sigma_n$, called the operation induced by σ. A morphism of categorical Σ-algebras is a functor which preserves the operations up to natural isomorphism.

We say that a categorical Σ-algebra $\mathscr{A}$ is **continuous** if it has initial object and colimits of ω-diagrams, moreover, the operations $\sigma^{\mathscr{A}}$ are continuous, i.e., they preserve colimits of ω-diagrams in each argument. Morphisms of continuous categorical Σ-algebras are continuous and preserve initial objects.

The notion of continuous categorical Σ-algebra generalizes the notion of continuous ordered Σ-algebra [14, 15], where the underlying category is a poset. Some examples of continuous categorical Σ-algebras are given below.

2.2 Linear Orderings

In this paper, a linear ordering $(W,<)$ is a **countable** set W equipped with a strict linear order relation $<$. (To force the collection of all words to be a small set, we may require that the underlying set of a linear ordering is a subset of a fixed set.) A **morphism** between linear orderings $(W,<) \to (V,<)$ is a function $W \to V$ which preserves the order relation (and is thus injective). The category **Lin** of linear orderings has as initial object the empty linear ordering denoted $\mathbf{0}$. Moreover, **Lin** has colimits of all ω-diagrams.

Let Δ be a ranked alphabet with $\Delta_2 = \{+\}$, $\Delta_0 = \{\mathbf{1}\}$ and $\Delta_n = \emptyset$ for all $n \notin \{0,2\}$. We turn **Lin** into a categorical Δ-algebra by interpreting the binary symbol $+$ as the usual **sum functor** $\mathbf{Lin}^2 \to \mathbf{Lin}$ and $\mathbf{1}$ as a singleton linear ordering. The sum functor maps a pair of linear orderings $(W_i,<_i)$, $i = 1,2$ to the linear ordering $(W_1 + W_2, <)$ whose underlying set is the disjoint union of W_1 and W_2 and such that the restriction of $<$ to W_i is $<_i$, for $i = 1,2$. The sum $h_1 + h_2$ of morphisms $h_i : W_i \to V_i$, $i = 1,2$ is defined so that it agrees with h_i on W_i, for $i = 1,2$. Equipped with these functors, **Lin** is a continuous categorical Δ-algebra.

2.3 Trees

Let Σ be any ranked set. An example of a continuous categorical Σ-algebra is the algebra T_Σ^∞ of all finite and infinite Σ-trees defined in the usual manner. This continuous categorical Σ-algebra is ordered, so that there is at most one morphism between any two trees. It is known that T_Σ^∞ is the *essentially unique* initial continuous categorical Σ-algebra. See [14, 15, 6] for more details.

3 Recursion Schemes

Definition 3.1. *A **recursion scheme** over Σ is a sequence E of equations*

$$\begin{array}{rcl} F_1(v_1,\ldots,v_{k_1}) & = & t_1 \\ & \vdots & \\ F_n(v_1,\ldots,v_{k_n}) & = & t_n \end{array} \tag{1}$$

*where t_i is a term over the ranked alphabet $\Sigma \cup \mathscr{F}$ in the variables $v_1,\ldots,v_{k_i}$, for $i \in [n]$, where $\mathscr{F} = \{F_1,\ldots,F_n\}$. A recursion scheme is **regular** if $k_i = 0$, for each $i \in [n]$.*

In the above definition, $\Sigma \cup \mathscr{F}$ is the ranked alphabet whose letters are the letters in Σ together with the letters in $\{F_1,\ldots,F_n\}$ where each F_i is of rank k_i.

In any continuous categorical Σ-algebra $\mathscr{A}$, any scheme E induces a continuous endofunctor $E^{\mathscr{A}}$ over the category

$$[\mathscr{A}^{k_1} \to \mathscr{A}] \times \ldots [\mathscr{A}^{k_n} \to \mathscr{A}]$$

where $[\mathscr{A}^k \to \mathscr{A}]$ denotes the category of all continuous functors $\mathscr{A}^k \to \mathscr{A}$. Since this category also has initial object and colimits of ω-diagrams, it has an **initial fixed point** $|E^{\mathscr{A}}|$ (cf. [1, 19]) which is unique up to isomorphism.

Definition 3.2. *Suppose that $\mathscr{A}$ is a continuous categorical Σ-algebra. We call a functor $f : \mathscr{A}^m \to \mathscr{A}$, **algebraic** if there is a recursion scheme E such that f is isomorphic to $|E|_1^A$, the first component of the above initial solution. When $m = 0$, f may be identified with an object of A, called an **algebraic object**. An object a in $\mathscr{A}$ is **regular** if there is a regular recursion scheme E such that a is isomorphic to $|E|_1^{\mathscr{A}}$.*

By applying the above notion to **Lin** and T_Σ^∞, we obtain the notions of algebraic and regular linear orderings, and algebraic and regular trees, respectively. Several characterizations of algebraic and regular trees can be found in [14, 10, 15]. For characterizations of regular linear orderings we refer to [9, 3]. Here we only mention the following fact.

Let Δ be the ranked alphabet defined above in Section 2.2. Then there is a unique morphism of categorical Δ-algebras $T_\Delta^\infty \to$ **Lin**, namely the **frontier map** mapping each tree to the linear ordering of its leaves. Due to a Mezei-Wright type result [6] we have:

Proposition 3.3. *A countable linear ordering is algebraic or regular iff it is isomorphic to the frontier of an algebraic or regular tree in T_Δ^∞.*

Actually the above fact holds for all ranked sets Σ such that Σ_0 is not empty and there is at least one $n > 1$ such that Σ_n is also not empty.

4 Representing Linear Orderings by Languages of Finite Words

Let A be an alphabet equipped with a fixed linear order relation that we extend to the lexicographic order $<_\ell$ of A^*, the set of (isomorphism types) of finite words. If $L \subseteq A^*$ is any language, then $(L, <_\ell)$ is a linear ordering. When A has two or more letters, then every countable linear ordering is isomorphic to a linear ordering $(L, <_\ell)$. We can also show that every **recursive** linear ordering is isomorphic to an ordering $(L, <_\ell)$, for some recursive language $L \subseteq A^*$.

Definition 4.1. *Call a linear ordering **context-free** (**deterministic context-free**, respectively) if it is isomorphic to a linear ordering $(L, <_\ell)$ for some context-free (deterministic context-free, respectively) language L over some alphabet A (or equivalently, over the 2-letter alphabet $\{0,1\}$).*

Using Courcelle's characterization of algebraic trees by deterministic context-free languages from [10] together with Proposition 3.3, we have:

Proposition 4.2. *A linear ordering is algebraic iff it is deterministic context-free.*

There is a similar characterization of regular linear orderings using ordinary regular languages.

5 Scattered Algebraic Linear Orderings

A good treatment of linear orderings is [17]. Recall from [17] that a linear ordering is **scattered** if it has no subordering isomorphic to the ordering of the rationals.

Scattered (countable) linear orderings can be classified into a transfinite hierarchy. Let V_0 denote the empty linear ordering and the singleton linear orderings. When α is a nonzero ordinal, let V_α be the collection of all linear orderings that can be obtained from a subordering P of $\mathbb{Z}$, the ordering of the integers by replacing each point $x \in P$ with a linear ordering in V_{β_x} for some $\beta_x < \alpha$. By Hausdorff's theorem, a linear ordering is scattered iff it belongs to V_α for some (countable) ordinal α, and the least such ordinal is called the **Hausdorff rank** or **VD**-rank of the scattered linear ordering.

It is known (see [16, 2, 5]) that a well-ordering is regular iff its order type is less than ω^ω, or equivalently, when its Hausdorff rank is finite. Moreover, the Hausdorff rank of every scattered regular linear ordering is finite. In [7], it is shown that a well-ordering is algebraic iff its order type is less than that of the ordinal ω^{ω^ω}, or equivalently, when its Hausdorff rank is less than ω^ω.

The main result of this paper is:

Theorem 5.1. *The Hausdorff rank of any scattered algebraic linear ordering is less than ω^ω.*

Corollary 5.2. *The Hausdorff rank of any scattered deterministic context-free ordering is less than ω^ω.*

6 Conclusion and Open Problems

A hierarchy of recursion schemes was introduced in [11], see also [12, 13]. Here, we dealt with level 0 (regular schemes) and level 1 (algebraic or first-order schemes) of the hierarchy. In Theorem 5.1, we have shown that every scattered linear ordering definable by a level 1 scheme is of Hausdorff rank less than ω^ω, whereas it has been known that the Hausdorff rank of any scattered linear ordering definable by a recursion scheme of level 0 is less than ω. We conjecture that for each n, the Hausdorff rank of any scattered linear ordering definable by a level n scheme is less than $\Uparrow(\omega, n+1)$, a tower of $n+1$ ω's. If that conjecture is true, then it follows that an ordinal is definable by a level n scheme iff it is less than $\Uparrow(\omega, n+2)$, and thus an ordinal is definable in the hierarchy iff it is less than ε_0. (See also [8].)

In ordinal analysis of logical theories, the strength of a theory is measured by ordinals. For example, the proof theoretic ordinal of Peano arithmetic is ε_0. Here we have a similar phenomenon: we measure the strength of recursive definitions by ordinals, and we conjecture that the ordinals definable are exactly those less than ε_0.

Finally, we mention two open problems.

Problem 1. Is there a context-free linear order which is not a deterministic context-free linear order?

Problem 2. Characterize the context-free well orderings and scattered linear orderings.

References

[1] J. Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolinae*, 15:589–602, 1974.

[2] S. L. Bloom and C. Choffrut. Long words: the theory of concatenation and ω-power. *Theor. Comput. Sci.*, 259(1–2):533–548, 2001.

[3] S. L. Bloom and Z. Ésik. Deciding whether the frontier of a regular tree is scattered. *Fund. Inform.*, 11:1–22, 2004.

[4] S. L. Bloom and Z. Ésik. The equational theory of regular words. *Inform. and Comput.*, 197(1–2):55–89, 2005.

[5] S. L. Bloom and Z. Ésik. Regular and algebraic words and ordinals. In T. Mossakowski et al., ed., *Proc. of 2nd Int. Conf. on Algebra and Coalgebra in Computer Science, CALCO 2007 (Bergen, Aug. 2007)*, v. 4624 of *Lect. Notes in Comput. Sci.*, pp. 1–15. Springer, 2007.

[6] S. L. Bloom and Z. Ésik. A Mezei-Wright theorem for categorical algebras. *Theor. Comput. Sci.*, to appear.

[7] S. L. Bloom and Z. Ésik. Algebraic ordinals. Submitted for publication.

[8] L. Braud. Unpublished paper.

[9] B. Courcelle. Frontiers of infinite trees. *Theor. Inform. and Appl.*, 12:319–337, 1978.

[10] B. Courcelle. Fundamental properties of infinite trees. *Theor. Comput. Sci.*, 25:95–169, 1983.

[11] W. Damm. Higher type program schemes and their tree languages. In H. Tzschach et al., eds., *Proc. of 3rd GI Conf. on Theoretical Computer Science (Darmstadt, March 1977)*, v. 48 of *Lect. Notes in Comput. Sci.*, pp. 51–72. Springer, 1977.

[12] W. Damm. The IO and OI hierarchies. *Theor. Comput. Sci.*, 20:95–206, 1982.

[13] J. Gallier. *n*-rational algebras I: basic properties and free algebras. *SIAM J. on Comput.*, 13:750–775, 1984.

[14] J. A. Goguen, J. W. Thatcher, E. G. Wagner and J. B. Wright. Initial algebraic semantics and continuous algebras. *J. of ACM*, 24:68–95, 1977.

[15] I. Guessarian. *Algebraic Semantics*, v. 99 of *Lect. Notes in Comput. Sci.* Springer, 1981.

[16] S. Heilbrunner. An algorithm for the solution of fixed-point equations for infinite words. *Theor. Inform. and Appl.*, 14:131–141, 1980.

[17] J. B. Rosenstein. *Linear Orderings*. Academic Press, 1982.

[18] W. Thomas. On frontiers of regular trees. *Theor. Inform. and Appl.*, 20:371–381, 1986.

[19] M. Wand. Fixed point constructions in order-enriched categories. *Theor. Comput. Sci.*, 8:13–30, 1979.

An Easy Completeness Proof for the Modal μ-Calculus on Finite Trees

Balder ten Cate[*]
INRIA Saclay - Île-de-France and ENS Cachan
61 avenue du President Wilson, F-94235 Cachan Cedex, France
balder.tencate@gmail.com
Gaëlle Fontaine[†]
Institute for Logic, Language and Computation, University of Amsterdam
P.O. Box 94242, NL-1090 GE Amsterdam, The Netherlands
gaelle.fontaine@uva.nl

The μ-calculus is an extension of modal logic with a fixpoint operator. In 1983, Dexter Kozen suggested an axiomatization (see, e.g., [4]). It took more than ten years to prove completeness. This proof is due to Igor Wałukiewicz [7] and is quite involved. We propose here a simpler proof in a particular case. More precisely, we prove the completeness of the Kozen axiomatization $\mathbf{K}^\mu$ extended with the axiom $\mu x.\Box x$ with respect to the class of finite tree models.

Our argument basically consists of three steps. The first step consist of defining a notion of rank which plays the same role as the modal depth for modal formulas. One of the main properties of the rank is the following. In order to know whether a formula φ of rank n is true at a node w, it is enough to know which proposition letters are true at w and which formulas of rank at most n are true at the successor nodes of w. Another key property of the rank is that there are only finitely many formulas of a given rank (up to logical equivalence).

The second step is to prove completeness of the μ-calculus with respect to generalized models, which are basically Kripke models augmented with a set of admissible subsets, in the style of Henkin semantics for second order logic. We do this by a standard canonical model construction.

The last step is inspired by a work of Kees Doets (see, e.g., [1]). Let us call a node in a generalized model n-good if there is a node in a finite tree model which satisfies exactly the same formulas of rank at most n. Using an induction principle, we show that every node in a generalized model satisfying $\mu x.\Box x$ is n-good. It is here that we use the main property of the rank. Finally, putting this together with the completeness for generalized models, we obtain completeness for the class of finite tree models.

This argument can also be applied to some extensions of the logic $\mathbf{K}^\mu + \mu x.\Box x$. More precisely, we show that when we add finitely many shallow axioms (as defined in [6]), we obtain a complete axiomatization for the corresponding class of finite trees. We also mention that we can adapt our proof to show completeness for the graded μ-calculus extended with the axiom $\mu x.\Box x$.

The paper is organized as follows. In section 1, we recall what is the Kozen axiomatization for the μ-calculus $\mathbf{K}^\mu$ and what is the intended semantics. In section 2, we define the notion of rank for a formula. In section 3, we give a definition for the generalized models and we show completeness of $\mathbf{K}^\mu$ with respect to the class of generalized models. In section 4, we use Kees Doets' argument to obtain completeness of $\mathbf{K}^\mu + \mu x.\Box x$ with respect to the class of finite tree models. In the last two sections, we give some examples of extensions of $\mathbf{K}^\mu + \mu x.\Box x$ to which we can apply our method in order to prove completeness.

[*] supported by the Netherlands Organization for Scientific Research (NWO), under grant 639.021.508 and by ERC Advanced Grant Webdam on Foundation of Web data management

[†] supported by VICI grant 639.073.501 of the NWO

1 Syntax, Semantics and Axiomatization

We introduce the language and the Kripke semantics for the μ-calculus. We also recall the axiomatization given by Dexter Kozen.

Definition 1.1. The μ-formulas over a set *Prop* of proposition letters are given by

$$\varphi ::= \top \mid p \mid x \mid \varphi \vee \varphi \mid \neg\varphi \mid \Diamond\varphi \mid \mu x.\varphi,$$

where p ranges over the set *Prop* and x ranges over the set *Var* of variables. In $\mu x.\varphi$, we require that the variable x appears only under an even number of negations in φ. We will assume that *Var* is infinite.

As usual, we let $\phi \wedge \psi$, $\Box\varphi$ and $\nu x.\varphi$ be abbreviations for $\neg(\neg\varphi \vee \neg\psi)$, $\neg\Diamond\neg\varphi$ and $\neg\mu x.\neg[\neg x/x]$.

The notions of *subformula*, *bound variable*, *free variable* and *substitution* are defined in the usual way. If φ and ψ are μ-formulas and if p is a proposition letter, we denote by $\varphi[\psi/p]$ the formula obtained by replacing in φ each occurrence of p by ψ. Similarly, if x is a variable, we define $\varphi[\psi/x]$.

A *μ-sentence* is a formula in which all the variables are bound.

Definition 1.2. A *Kripke frame* is a pair (W,R), where W is a set and R a binary relation on W. A *Kripke model* is a triple (W,R,V) where (W,R) is a Kripke frame and $V : Prop \to \mathscr{P}(W)$ a valuation. If (w,v) belongs to R, we say that w is a *predecessor* of v and v is a *successor* of w.

Given a formula φ, a Kripke model $\mathscr{M} = (W,R,V)$ and an assignment $\tau : Var \to \mathscr{P}(W)$, we define a subset $[\![\varphi]\!]_{\mathscr{M},\tau}$ that is interpreted as the set of points at which φ is true. The subset is defined by induction in the usual way. We only recall that

$$[\![\mu x.\varphi]\!]_{\mathscr{M},\tau} = \bigcap\{U \subseteq W : [\![\varphi]\!]_{\mathscr{M},\tau[x:=U]} \subseteq U\},$$

where $\tau[x := U]$ is the assignment τ' such that $\tau'(x) = U$ and $\tau'(y) = \tau(y)$, for all $y \neq x$. Observe that the set $[\![\mu x.\varphi]\!]_{\mathscr{M},\tau}$ is the least fixpoint of the map $\varphi_x : \mathscr{P}(W) \to \mathscr{P}(W)$ defined by $\varphi_x(U) := [\![\varphi]\!]_{\mathscr{M},\tau[x:=U]}$, for all $U \subseteq W$.

If $w \in [\![\varphi]\!]_{\mathscr{M},\tau}$, we write $\mathscr{M}, w \Vdash_\tau \varphi$ and we say that φ is *true* at w under the assignment τ. If φ is a sentence, we simply write $\mathscr{M}, w \Vdash \varphi$.

A formula φ is *true* in $\mathscr{M}$ under an assignment τ if for all $w \in W$, we have $\mathscr{M}, w \Vdash_\tau \varphi$. In this case, we write $\mathscr{M} \Vdash_\tau \varphi$. A set Φ of formulas is *true* in a model $\mathscr{M}$ under an assignment τ, notation: $\mathscr{M} \Vdash_\tau \Phi$, if for all φ in Φ, φ is true in $\mathscr{M}$ under τ.

Finally, if (W,R) is a Kripke frame and for all valuations V and all assignments τ, φ is true in (W,R,V) under the assignment τ, we say that φ is *valid* in (W,R) and we write $(W,R) \Vdash \varphi$.

Definition 1.3. The axiomatization of the Kozen system $\mathbf{K}^\mu$ consists of the following axioms and rules

propositional tautologies,	
If $\vdash \varphi \to \psi$ and $\vdash \varphi$, then $\vdash \psi$	(Modus ponens),
If $\vdash \varphi$, then $\vdash \varphi[p/\psi]$	(Substitution),
$\vdash \Box(p \to q) \to (\Box p \to \Box q)$	(*K*-axiom),
If $\vdash \varphi$, then $\vdash \Box\varphi$	(Necessitation),
$\vdash \varphi[x/\mu x.\varphi] \to \mu x.\varphi$	(Fixpoint axiom),
If $\vdash \varphi[x/\psi] \to \psi$, then $\vdash \mu x.\varphi \to \psi$	(Fixpoint rule),

where x is not a bound variable of φ and no free variable of ψ is bound in φ.

Definition 1.4. If Φ is a set of μ-formulas, we write $\mathbf{K}^\mu + \Phi$ for the smallest set of formulas which contains both $\mathbf{K}^\mu$ and Φ and is closed for the Modus Ponens, Substitution, Necessitation and Fixpoint rules.

Definition 1.5. Let (W,R) be a Kripke frame. A point r in W is a *root* if for all w in W, there is a sequence $w_0,\dots,w_n$ such that $w_0 = r$, $w_n = w$ and (w_i,w_{i+1}) belongs to R, for all $i \in \{0,\dots,n-1\}$.

The frame (W,R) is a *tree* if it has a root, every point distinct from the root has a unique predecessor and there is no sequence $w_0,\dots,w_{n+1}$ in W such that $w_{n+1} = w_0$ and (w_i,w_{i+1}) belongs to R, for all $i \in \{0,\dots,n\}$. The frame (W,R) is a *finite tree* if it is a tree and W is finite.

Finally, a *finite tree Kripke model* is a Kripke model (W,R,V) such that (W,R) is a finite tree.

Fact 1.6. *Let $\mathscr{M} = (W,R,V)$ be a Kripke model. The formula $\mu x.\Box x$ is true at a point w in $\mathscr{M}$ iff there is no infinite sequence $w_0,w_1\dots$ in W such that $w_0 = w$ and (w_i,w_{i+1}) belongs to R, for all $i \in \mathbb{N}$.*

In particular, the formula $\mu x.\Box x$ is true in $\mathscr{M}$ iff there is no infinite sequence $w_0,w_1,\dots$ such that (w_i,w_{i+1}) belongs to R, for all $i \in \mathbb{N}$. That is, iff $\mathscr{M}$ is conversely well-founded.

We prove the completeness of the logic $\mathbf{K}^\mu + \mu x.\Box x$ with respect to the class of finite tree Kripke models. That is, a formula φ is provable in $\mathbf{K}^\mu + \mu x.\Box x$ iff it is valid in any finite tree Kripke model. Note that this result can be easily derived from the completeness result proved by Igor Wałukiewicz in [7].

2 Rank of a Formula

The goal of this section is to come up with a definition of rank that would be the analogue of the depth of a modal formula. For modal logic, it is not hard to see that the truth of an arbitrary formula φ at some world w only depends of the truth of the proposition letters at w and of the truth of formulas ψ at the successors of w, where the depth of ψ is at most the depth of φ. In our proof, we will need something similar for the μ-calculus.

The most natural idea would be to look at the nesting depth of modal and fixpoint operators. However, this definition does not have the required properties. The notion of rank that we develop in this section is in fact related to the closure of a formula, which has been introduced by Dexter Kozen in [4].

Definition 2.1. The *closure* $Cl(\varphi)$ of a formula φ is the smallest set of formulas such that

$$\begin{array}{l}
\varphi \in Cl(\varphi),\\
\text{if } \Diamond\psi \in Cl(\varphi), \text{ then } \psi \in Cl(\varphi),\\
\text{if } \neg\psi \in Cl(\varphi), \text{ then } \psi \in Cl(\varphi),\\
\text{if } \mu x.\psi \in Cl(\varphi), \text{ then } \psi[x/\mu x.\psi] \in Cl(\varphi).\\
\text{if } \psi \vee \chi \in Cl(\varphi), \text{ then both } \psi,\chi \in Cl(\varphi),
\end{array}$$

It is also proved in [4] that the closure $Cl(\varphi)$ of a formula φ is finite. In order to define the rank, we also need to recall the notion of the depth of a formula.

Definition 2.2. The *depth* $d(\varphi)$ of a formula φ is defined by induction as follows

$$\begin{array}{l}
d(\top) = d(p) = d(x) = 0,\\
d(\varphi \vee \psi) = max\{d(\varphi),d(\psi)\},\\
d(\neg\varphi) = d(\varphi),\\
d(\Diamond\varphi) = d(\mu x.\varphi) = d(\varphi)+1.
\end{array}$$

Definition 2.3. The *rank* of a formula φ is defined as follows

$$rank(\varphi) = max\{d(\psi) \mid \psi \in Cl(\varphi)\}.$$

Remark that since $Cl(\varphi)$ is finite, $rank(\varphi)$ is always a natural number. All we will use later are the following properties of the rank.

Proposition 2.4. *If the set Prop of proposition letters is finite, then for all natural numbers k, there are only finitely many sentences of rank k (up to logical equivalence).*

Proof. Fix a natural number k. Note first that if $rank(\varphi) = k$, then in particular, $d(\varphi) \leq k$. Hence, it is enough to show that there only finitely many sentences of depth below k (up to logical equivalence). If $d(\varphi) \leq k$, we may assume that the only variables occurring in φ are some $x_1,\dots,x_k$. It is routine to prove by induction on l that there are finitely many formulas of depth l with variables $x_1,\dots,x_k$. □

Proposition 2.5. *The rank is closed under boolean combination. That is, for any n, a boolean combination of formulas of rank at most n is a formula of rank at most n.*

Proposition 2.6. *Every formula φ is provably equivalent to a boolean combination of proposition letters and formulas of the form $\Diamond\psi$, with $rank(\psi) \leq rank(\varphi)$.*

Proof. Recall that a formula is guarded if every bound variable is in the scope of a modal operator. It can be shown that every formula is provably equivalent to a guarded formula. Therefore, let φ be a guarded formula. We define a map G by induction as follows:

$$\begin{array}{rcl}
G(\top) & = & \top, \\
G(p) & = & p, \text{ if } p \text{ is a free variable of } \varphi, \\
G(\neg\psi) & = & \neg G(\psi), \\
G(\psi \vee \psi') & = & G(\psi) \vee G(\psi'), \\
G(\Diamond\psi) & = & \Diamond\psi, \\
G(\mu x.\psi) & = & G(\psi[x/\mu x.\psi]).
\end{array}$$

Note that G is not defined for a bound variable x of φ. Using the fact that φ is guarded, one can show that the computation of $G(\varphi)$ is well-defined and does terminate. It is not hard to see that $G(\varphi)$ is equivalent to φ. Remark now that if ψ belongs to $Cl(\varphi)$, then $Cl(\psi)$ is a subset of $Cl(\varphi)$. It follows that $G(\varphi)$ is a boolean combination of proposition letters and formulas of the form $\Diamond\psi$, with $rank(\psi) \leq rank(\varphi)$. □

3 Completeness for Generalized Models

We introduce generalized models which are the analogue for the μ-calculus of the general models for second order logic. We prove completeness of $\mathbf{K}^\mu$ with respect to the class of generalized models.

Definition 3.1. Consider a quadruple $\mathcal{M} = (W,R,V,\mathbb{A})$ where (W,R) is a Kripke frame, $\mathbb{A}$ is a subset of $\mathcal{P}(W)$ and $V : Prop \to \mathbb{A}$ a valuation. A set which belongs to $\mathbb{A}$ is called *admissible*.

We define the truth of a formula φ under an assignment $\tau : Var \to \mathbb{A}$ by induction. Remark that all the clauses are the same as usual, except the one defining the truth of $\mu x.\varphi$. Normally, we define the set $[\![\mu x.\varphi]\!]_{\mathcal{M},\tau}$ as the least pre-fixpoint of the map φ_x (see Definition 1.2). But here, we define it as the intersection of all the pre-fixpoints of φ_x, that are admissible.

$$\begin{array}{l}
[\![\top]\!]_{\mathcal{M},\tau} = W, \\
[\![p]\!]_{\mathcal{M},\tau} = V(p), \\
[\![x]\!]_{\mathcal{M},\tau} = \tau(x), \\
[\![\neg\varphi]\!]_{\mathcal{M},\tau} = W \backslash [\![\varphi]\!]_{\mathcal{M},\tau}, \\
[\![\varphi \vee \psi]\!]_{\mathcal{M},\tau} = [\![\varphi]\!]_{\mathcal{M},\tau} \cup [\![\psi]\!]_{\mathcal{M},\tau}, \\
[\![\Diamond\varphi]\!]_{\mathcal{M},\tau} = \{w \in W : \exists v \in W \text{ s.t. } wRv \text{ and } v \in [\![\varphi]\!]_{\mathcal{M},\tau}\}, \\
[\![\mu x.\varphi]\!]_{\mathcal{M},\tau} = \bigcap\{U \in \mathbb{A} : [\![\varphi]\!]_{\mathcal{M},\tau[x:=U]} \subseteq U\},
\end{array}$$

where $\tau[x := U]$ is the assignment τ' such that $\tau'(x) = U$ and $\tau(y) = \tau(y)$, for all $y \neq x$. If $w \in [\![\varphi]\!]_{\mathscr{M},\tau}$, we write $\mathscr{M}, w \Vdash_\tau \varphi$ and we say that φ is *true* at w under the assignment τ. If φ is a sentence, we simply write $\mathscr{M}, w \Vdash \varphi$. A formula φ is *true* in $\mathscr{M}$ under an assignment τ if for all $w \in W$, we have $\mathscr{M}, w \Vdash_\tau \varphi$. In this case, we write $\mathscr{M} \Vdash_\tau \varphi$.

The quadruple $\mathscr{M} = (W, R, V, \mathbb{A})$ is a *generalized model* if for all formulas φ and all assignments $\tau : Var \to \mathbb{A}$, the set $[\![\varphi]\!]_{\mathscr{M},\tau}$ belongs to $\mathbb{A}$. A triple $\mathscr{F} = (W, R, \mathscr{A})$ is a *generalized frame* if for every valuation $V : Prop \to \mathbb{A}$, the quadruple $(W, R, V, \mathbb{A})$ is a generalized model.

If $\mathscr{F} = (W, R, \mathscr{A})$ is a generalized frame, we call (W, R) the *underlying Kripke frame* of $\mathscr{F}$. A formula φ is *valid* in a generalized frame $\mathscr{F} = (W, R, \mathscr{A})$, notation: $\mathscr{F} \Vdash \varphi$, if for all valuations $V : Prop \to \mathbb{A}$ and all assignments $\tau : Var \to \mathbb{A}$, the formula φ is true in $(W, R, V, \mathbb{A})$ under the assignment τ.

Remark that any Kripke model $M = (W, R, V)$ can be seen as the generalized model $M' = (W, R, V, \mathscr{P}(W))$. It follows easily from our definition that for all formulas φ and all points $w \in W$,

$$M, w \Vdash \varphi \quad \text{iff} \quad M', w \Vdash \varphi.$$

Theorem 3.2. $\mathbf{K}^\mu$ *is complete with respect to the class of generalized models. That is, for any formula* φ*,* $\vdash_{\mathbf{K}^\mu} \varphi$ *iff for any generalized model* $\mathscr{M}$*,* $\mathscr{M} \Vdash \varphi$*.*

Proof. The argument is similar to the modal case and uses a variant of the standard canonical model construction (see, e.g.g, [5]). □

4 Completeness for Finite Tree Models

In the style of Kees Doets [1], we prove completeness of $\mathbf{K}^\mu + \mu x.\Box x$ with respect to the class of finite tree Kripke models. The argument is as follows. First, we say that a point w in a generalized model is n-good if there is a point v in a finite tree Kripke model such that no formula of rank at most n can distinguish w from v. Next, we show that "being n-good" is a property that can be expressed by a formula γ_n of rank at most n. Afterwards, we prove that each point (in a generalized model) satisfying $\mu x.\Box x$, is n-good. Finally, using completeness for generalized models, we obtain completeness of $\mathbf{K}^\mu + \mu x.\Box x$ with respect to the class of finite tree Kripke models.

In this section, we will assume that the set *Prop* of proposition letters is finite. Often we write "finite tree" instead of "finite tree Kripke model".

Definition 4.1. Fix a natural number n. Let $\mathscr{M}$ and $\mathscr{M}'$ be two generalized models. A world $w \in \mathscr{M}$ is *rank n-indistinguishable* to a world $w' \in \mathscr{M}'$ if for all formulas φ of rank at most n, we have

$$\mathscr{M}, w \Vdash \varphi \quad \text{iff} \quad \mathscr{M}', w' \Vdash \varphi.$$

In case this happens, we write $(\mathscr{M}, w) \sim_n (\mathscr{M}', w')$. Finally, we say that $w \in \mathscr{M}$ is *n-good* if there exists a finite tree $\mathscr{N}$ and some $v \in \mathscr{N}$ such that $(\mathscr{M}, w) \sim_n (\mathscr{N}, v)$.

Definition 4.2. Let n be a natural number and let Φ_n be the set of formulas of rank at most n. For any generalized model $\mathscr{M}$ and any $w \in \mathscr{M}$, we define the *n-type* $\theta_n(w)$ as the set of formulas in Φ_n which are true at w.

Remark that by Proposition 2.4, Φ_n is finite (up to logical equivalence) and in particular, there are only finitely many distinct n-types.

Lemma 4.3. *Let n be a natural number. There exists a formula* γ_n *of rank n such that for any generalized model* $\mathscr{M}$ *and any* $w \in \mathscr{M}$*, we have*

$$\mathscr{M}, w \Vdash \gamma_n \quad \textit{iff} \quad (\mathscr{M}, w) \textit{ is n-good.}$$

Proof. Let n be a natural number and let γ_n be the formula defined by

$$\gamma_n = \bigvee\{\bigwedge \theta_n(w) \mid w \text{ is } n\text{-good}\},$$

where w a point in a generalized model $\mathscr{M}$ and $\bigwedge \theta_n(w)$ is shorthand for $\bigwedge\{\varphi : \varphi \in \theta_n(w)\}$. Note that since there are only finitely many distinct n-types, the formula γ_n is well-defined. Moreover, from Proposition 2.5, it follows that the rank of γ_n is n.

It remains to check that γ_n has the required properties. It is immediate to see that if a point w in a generalized model is n-good, then γ_n is true at w. For the other direction, assume that γ_n is true at a point w in a generalized model $\mathscr{M}$. Therefore, there is a point w' in a generalized model $\mathscr{M}'$ such that w' is n-good and $\theta_n(w')$ is true at w. Since w' is n-good, there is a point v in a model $\mathscr{N}$ such that w' and v are rank n-indistinguishable. Using the fact that w and w' have the same n-type, we obtain that w and v are also rank n-indistinguishable. That is, w is n-good. □

Lemma 4.4. *For all natural numbers n, $\vdash_{\mathbf{K}^\mu} \Box\gamma_n \to \gamma_n$.*

Proof. Let n be a natural number. By Theorem 3.2, it is enough to show that the formula $\Box\gamma_n \to \gamma_n$ is valid in all generalized models. Let $\mathscr{M}$ be a generalized model and let $w \in \mathscr{M}$. We have to show $\mathscr{M}, w \Vdash \Box\gamma_n \to \gamma_n$. So suppose $\mathscr{M}, w \Vdash \Box\gamma_n$. If w is a reflexive point, we immediately obtain $\mathscr{M}, w \Vdash \gamma_n$ and this finishes the proof. Assume now that w is irreflexive. We have to prove that $(\mathscr{M}, w)$ is n-good. That is, we have to find a finite tree $\mathscr{N}$ and some $v \in \mathscr{M}$ such that $(\mathscr{M}, w) \sim_n (\mathscr{N}, v)$.

Now for any successor u of w, we have $\mathscr{M}, u \Vdash \gamma_n$. Therefore, $(\mathscr{M}, u)$ is n-good and there exists a finite tree $\mathscr{M}_u = (W_u, R_u, V_u)$ and some $w_u \in W_u$ such that $(\mathscr{M}, u) \sim_n (\mathscr{M}_u, w_u)$. Without loss of generality, we may assume that w_u is the root of $\mathscr{M}_u$.

The idea is now to look at the disjoint union of these models and to add a root v (that would be rank n-indistinguishable from w). However, this new model might not be a finite tree (w might have infinitely many successors). The solution is to restrict ourselves to finitely many successors of w. More precisely, for each n-type, we pick at most one successor of w.

So let U be a set of successors of w such that for any successor u of w, there is exactly one point u' of U satisfying $\theta_n(u) = \theta_n(u')$. Remark that since there are only finitely many distinct n-types, U is finite. Let $\mathscr{N} = (W, R, V)$ be the model defined by

$$\begin{aligned} W &= \{v\} \cup \biguplus\{W_u : u \in U\}, \\ R &= \{(v, w_u) : u \in U\} \cup \bigcup\{R_u : u \in U\}, \\ V(p) &= \begin{cases} \{v\} \cup \bigcup\{V_u(p) : u \in U\} & \text{if } \mathscr{M}, w \Vdash p, \\ \bigcup\{V_u(p) : u \in U\} & \text{otherwise,} \end{cases} \end{aligned}$$

for all proposition letters p. Since U is finite, $\mathscr{N}$ is a finite tree. Thus, it is enough to check that for any formula φ of rank at most n, we have

$$\mathscr{M}, w \Vdash \varphi \quad \text{iff} \quad \mathscr{N}, v \Vdash \varphi.$$

By Proposition 2.6, φ is provably equivalent to a boolean combination of proposition letters and formulas of the form $\Diamond\psi$, where $rank(\psi)$ is at most n. Thus, it is enough to show that w and v satisfy exactly the same proposition letters and the same formulas $\Diamond\psi$ with $rank(\psi) \leq n$.

By definition of V, it is immediate that w and v satisfy the same proposition letters. Now let ψ be a formula of rank at most n. We have to show that

$$\mathscr{M}, w \Vdash \Diamond\psi \quad \text{iff} \quad \mathscr{N}, v \Vdash \Diamond\psi.$$

For the direction from left to right, suppose that $\mathscr{M},w \Vdash \Diamond\psi$. Thus, there exists a successor u of w such that $\mathscr{M},u \Vdash \psi$. By definition of U, there is $u' \in U$ such that $(\mathscr{M},u) \sim_n (\mathscr{M},u')$. Thus, $(\mathscr{M},u) \sim_n (\mathscr{M}_{u'},w_{u'})$ and in particular, $\mathscr{M}_{u'},w_{u'} \Vdash \psi$. By definition of R, it follows that $\mathscr{N},v \Vdash \Diamond\psi$. The direction from right to left is similar. □

Proposition 4.5. *For all natural numbers n, $\vdash_{\mathbf{K}^\mu} \mu x.\Box x \to \gamma_n$.*

Proof. By Lemma 4.4, we know that $\Box\gamma_n \to \gamma_n$ is provable in $\mathbf{K}^\mu$. By the Fixpoint rule, we obtain that $\mu x.\Box x \to \gamma_n$ is provable in $\mathbf{K}^\mu$. □

Theorem 4.6. $\mathbf{K}^\mu + \mu x.\Box x$ *is complete with respect to the class of finite tree Kripke models.*

Proof. For any finite tree $\mathscr{M}$, we have $\mathscr{M} \Vdash \mathbf{K}^\mu$ and $\mathscr{M} \Vdash \mu x.\Box x$. Thus, it is sufficient to show that if φ is not provable in $\mathbf{K}^\mu + \mu x.\Box x$, there exists a finite tree N such that $N \nVdash \varphi$. Let φ be such a formula. In particular, $\nvdash_{\mathbf{K}^\mu} \mu x.\Box x \to \varphi$. By Theorem 3.2, we have $\mathscr{M},w \nVdash \mu x.\Box x \to \varphi$, for some generalized model $\mathscr{M}$ and some $w \in \mathscr{M}$.

Let n be the rank of φ. By Theorem 3.2 and Proposition 4.5, we get that $\mathscr{M},w \Vdash \mu x.\Box x \to \gamma_n$. Since $\mathscr{M},w \Vdash \mu x.\Box x$, it follows that $\mathscr{M},w \Vdash \gamma_n$. Therefore, there exists a finite tree $\mathscr{N}$ and some $v \in \mathscr{N}$ such that $(\mathscr{M},w) \sim_n (\mathscr{N},v)$. Since $\mathscr{M},w \nVdash \varphi$, we have $\mathscr{N},v \nVdash \varphi$. □

As mentioned before, this result also follows from the completeness of $\mathbf{K}^\mu$ showed by Igor Wałukiewicz in [7]. We briefly explain how to derive Theorem 4.6 from the completeness of $\mathbf{K}^\mu$. Recall that in [7], Igor Walukiewicz showed that a sentence φ is provable in $\mathbf{K}^\mu$ iff it is valid in all trees.

Suppose that a sentence φ is not provable in $\mathbf{K}^\mu + \mu x.\Box x$. In particular, the formula $\mu x.\Box x \to \varphi$ is not provable in $\mathbf{K}^\mu$. It follows from the completeness of $\mathbf{K}^\mu$ that there is a model $\mathscr{M} = (W,R.V)$ and a point w in W such that (W,R) is a tree and $\mu x.\Box x \to \varphi$ is not true at w. We may assume that w is the root of (W,R).

Since $\mu x.\Box x$ is true at w and since w is the root, it follows from Fact 1.6 that the tree (W,R) is conversely well-founded. Let n be the rank of φ. Now, if a point v in W has more than one successor of a given n-type θ, we can pick one successor of n-type θ and delete all the other successors of n-type θ. This would not modify the fact that φ is not true at w. By doing this operation inductively and using the fact that (W,R) is well-founded, we can prove that the tree (W,R) may be assumed to be finite. Therefore, there is a finite tree (W,R) in which φ is not valid.

5 Adding Shallow Axioms to $\mathbf{K}^\mu + \mu x.\Box x$

By slightly modifying our method, it is also possible to prove that the logic obtained by adding the axiom $\Diamond p \to \Box p$ to $\mathbf{K}^\mu + \mu x.\Box x$ is complete with respect to the class of finite strings (recall that a string is a tree such that every point has at most one successor).

We do not provide the details of the proof but it consists in two parts. First, we show that if we construct a canonical generalized model for this logic, then the underlying Kripke frame satisfies the axiom $\Diamond p \to \Box p$. Second, we modify the definition of being n-good by requiring in Definition 4.1 that the model $\mathscr{N}$ is a finite string. Then we prove that the lemmas 4.3 and 4.4 still holds for this new definition.

Theorem 5.1. *The logic $\mathbf{K}^\mu + \mu x.\Box x + (\Diamond p \to \Box p)$ is complete with respect to the class of finite strings.*

We remark that this theorem follows from a result by Roope Kaivola (see, e.g., [3]). But the proof proposed here is simpler.

More generally, we can also show that when we extend the logic $\mathbf{K}^\mu + \mu x.\Box x$ with axioms that are shallow (defined below), we obtain complete axiomatizations for the corresponding class of finite trees.

Definition 5.2 ([6])**.** A formula is *shallow* if no occurrence of a proposition letter is in the scope of a fixpoint operator and each occurrence of a proposition letter is in the scope of at most one modality. In other words, the shallow formulas is the language defined by

$$\varphi ::= \psi \mid \Diamond\psi \mid \varphi \vee \varphi \mid \neg\varphi,$$

where ψ is either a formula without any proposition letter or a propositional formula (that is a formula of the μ-calculus that does not contain neither $\Diamond$ nor μ).

Observe that the formula $\Diamond p \to \Box p$ is a shallow formula. Other examples are formulas expressing that each point has at most two successors ($\Diamond p \wedge \Diamond(q \vee \neg p) \to \Box(p \vee q)$), or that each point has at most one blind successor ($\Diamond(p \wedge \Box\bot) \wedge \Box(\Box\bot \to p)$).

Recall that a formula φ defines a class $\mathscr{C}$ of finite trees if $\mathscr{C}$ is exactly the class of trees which make φ valid.

Theorem 5.3. *Let φ be a shallow formula. Then the logic $\mathbf{K}^{\mu} + \mu x.\Box x + \varphi$ is complete with respect to the class of finite trees defined by φ.*

The structure of the proof is similar to the one of the proof of Theorem 5.1. Here, in order to show that the underlying frame of the canonical generalized model satisfies φ, we use the fact that the shallow formulas are canonical, which was proved in [6].

6 Graded μ-Calculus

Finally, we would like to mention that we can also use the same method to show that we can obtain a complete axiomatization for the graded μ-calculus together with the axiom $\mu x.\Box x$. In [2], Maurizio Fattorosi-Barnaba and Claudio Cerrato gave an axiomatization of graded modal logic and show that this axiomatization was complete with respect to the class of frames. If we add the Fixpoint axiom, the Fixpoint rule and the axiom $\mu x.\Box x$ to their axiomatization, we obtain a logic that is complete with respect to the class of finite trees.

The only part of the proof which requires some extra work is when we want to show a result similar to Theorem 3.2. Indeed, the canonical construction for graded modal logic is already not very easy. In fact, in order to show completeness for graded μ-calculus with respect to the class of generalized frames, we use directly the completeness result by Maurizio Fattorosi-Barnaba and Claudio Cerrato, instead of going trough the canonical model construction. This is done by translating each μ-formula into a modal formula, but over a larger set of proposition letters. We do not give the details, by lack of space.

Acknowledgements

We would like to thank Alexandru Baltag and Yde Venema for their comments on earlier drafts.

References

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, v. 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 2001.

[2] B. ten Cate. *Model Theory for Extended Modal Languages*. PhD thesis, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01.

[3] K. Doets. Monadic Π^1_1-theories of Π^1_1 properties. *Notre Dame J. of Formal Log.*, 30(2):224–240, 1989.

[4] M. Fattorosi-Barnaba and C. Cerrato. Graded modalities III: the completeness and compactness of S4^0. *Studia Logica*, 47(2):99–110, 1988.

[5] R. Kaivola. *Using Automata to Characterise Fixed Point Temporal Logics.* PhD thesis, University of Edinburgh, 1997.

[6] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[7] I. Wałukiewicz. A note on the completeness of Kozen's axiomatization of the propositional μ-calculus. *Bull. of Symb. Log.*, 2(3):349–366, 1996.

Least and Greatest Fixpoints in Game Semantics

Pierre Clairambault
PPS, CNRS & Université Paris 7
175 rue du Chevaleret, F-75013 Paris, France
pierre.clairambault@pps.jussieu.fr

1 Introduction

The idea to model logic by game-theoretic tools can be traced back to the work of Lorenzen [19]. The idea is to interpret a formula by a game between two players O and P, O trying to refute the formula and P trying to prove it. The formula A is then valid if P has a *winning strategy* on the interpretation of A. Later, Joyal remarked [18] that it is possible to compose strategies in Conway games [10] in an associative way, thus giving rise to the first category of games and strategies. This, along with parallel developments in Linear Logic and Geometry of Interaction, led to the more recent construction of compositional game models for a large variety of logics [1, 21, 11] and programming languages [17, 3, 20, 2].

On the other hand, games with parity conditions [4] have been used accurately in order to model languages such as the propositional μ-calculus [22]. The idea is to build a μ-bicomplete category of games, *i.e.* a category with finite products, finite coproducts and initial algebras/terminal coalgebras for all functors definable with products, coproducts and parametrized initial/terminal algebras/coalgebras. However, this category of games is a bit unsatisfactory from the point of view of proof theory since it is not *closed*, *i.e.* it does not admit an interpretation of functional types or implication.

2 On Previous Work

In a previous paper [6], we presented a logic named μLJ which is an extension of both the propositional μ-calculus and of the intuitionistic sequent LJ[15]. Variants of this language have already been considered, see for example [5]. Basically, it consists of the usual rules of LJ plus the following rules to express fixpoints, and the action of formulas with a free type variable as covariant (denoted T) or contravariant (denoted N) endofunctors.

Rules for Fixpoints and Functors

$$\frac{\Gamma \vdash T[\mu X.T/X]}{\Gamma \vdash \mu X.T}\,\mu_r \qquad\qquad \frac{T[A/X] \vdash A}{\mu X.T \vdash A}\,\mu_l$$

$$\frac{T[\nu X.T/X] \vdash B}{\nu X.T \vdash B}\,\nu_l \qquad\qquad \frac{A \vdash T[A/X]}{A \vdash \nu X.T}\,\nu_r$$

$$\frac{A \vdash B}{T(A) \vdash T(B)}\,[T] \qquad\qquad \frac{B \vdash A}{N(A) \vdash N(B)}\,[N]$$

This logic is then equipped with the usual reduction rules of LJ, with the addition of the following rules for fixpoints, and rules for the expansion of functors.

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash T[\mu X.T/X]}}{\Gamma \vdash \mu X.T}\,\mu_r \qquad \dfrac{\dfrac{\pi_2}{T[A/X] \vdash A}}{\mu X.T \vdash A}\,\mu_l}{\Gamma \vdash A}\,Cut \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash T[\mu X.T/X]} \qquad \dfrac{\dfrac{\dfrac{\pi_2}{T[A/X] \vdash A}}{\mu X.T \vdash A}\,\mu_l}{T[\mu X.T/X] \vdash T[A/X]}\,[T]}{\Gamma \vdash T[A/X]}\,Cut \qquad \dfrac{\pi_2}{\Gamma, T[A/X] \vdash A}}{\Gamma \vdash A}\,Cut$$

Figure 1: Cut reduction for μ

In [6], we show how to build a games model of this logic in the setting of *arena games* [17, 16]. We start from McCusker's model of recursive types in arena games [20] where recursive types are obtained by infinite iteration of the functors, in the spirit of Knaster-Tarski's fixed point theorem. We first revisit his work by replacing this infinite iteration process by loops in arenas. For this purpose, we introduce a general form of functors in game semantics, called *open functors*, which are semantic counterparts of formulas with free type variables. Such functors are in one-to-one correspondence with *open arenas*, *i.e.* arenas with special distinguished moves called *holes*, representing type variables. These arenas admit a *loop construction*, giving rise to a *minimal invariant* [14, 12, 13] for the corresponding functor. Moreover, we show that this loop construction can be enriched by *parity winning conditions*, providing initial algebras and terminal coalgebras for most covariant open functors. Hence, we have built a category of games which is cartesian closed, has (weak) coproducts and initial algebras/terminal coalgebras for all covariant functors definable with the language constructors (including fixpoints and implication) : this is our model for μLJ.

Whereas this version of μLJ is interesting in its own right, there remains a drawback making it unsatisfactory for a plausible programming language with induction/coinduction, namely, the absence of context in the rules for μ/ν : this restriction would correspond to a programming language where one can only iterate a closed term. We are now interested in the following extended rules:

Extended Rules for Fixpoints and Functors

$$\dfrac{\Gamma \vdash T[\mu X.T/X]}{\Gamma \vdash \mu X.T}\,\mu_r \qquad \dfrac{\Gamma, T[A/X] \vdash A}{\Gamma, \mu X.T \vdash A}\,\mu_l$$

$$\dfrac{\Gamma, T[\nu X.T/X] \vdash B}{\Gamma, \nu X.T \vdash B}\,\nu_l \qquad \dfrac{\Gamma, A \vdash T[A/X]}{\Gamma, A \vdash \nu X.T}\,\nu_r$$

$$\dfrac{\Gamma, A \vdash B}{\Gamma, T(A) \vdash T(B)}\,[T] \qquad \dfrac{\Gamma, B \vdash A}{\Gamma, N(A) \vdash N(B)}\,[N]$$

While it is true that, as claimed in [6], these general rules can be derived from the previous ones, it is unclear and non-trivial whether these derivations are correct from the dynamical point of view. For example, one would have to show that the reduction presented in Figure 2 holds. Unfortunately, due to the complexity of the derivations for the extended rules, the required verifications turn out to be unfeasible, at least by hand. Hence, we instead take these extended rules and reductions as primitive, and investigate possible strengthenings of our games model which could validate them.

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma\vdash T[\mu X.T/X]}}{\Gamma\vdash \mu X.T}\,\mu_r \qquad \dfrac{\dfrac{\pi_2}{\Gamma,T[A/X]\vdash A}}{\Gamma,\mu X.T\vdash A}\,\mu_l}{\Gamma\vdash A}\,Cut \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma\vdash T[\mu X.T/X]} \qquad \dfrac{\dfrac{\dfrac{\pi_2}{\Gamma,T[A/X]\vdash A}}{\Gamma,\mu X.T\vdash A}\,\mu_l}{\Gamma,T[\mu X.T/X]\vdash T[A/X]}\,[T]}{\Gamma\vdash T[A/X]}\,Cut \qquad \dfrac{\pi_2}{\Gamma,T[A/X]\vdash A}}{\Gamma\vdash A}\,Cut$$

Figure 2: Extended cut reduction for μ

3 A Categorical Setting of Strong Functors

We first investigate what categorical structure is needed in order to interpret the extended rules. Let us suppose given a cartesian closed category $\mathscr{C}$, with (weak) coproducts. In this section, we will be interested in the notion of *strong endofunctors*. These are defined as functors $T : \mathscr{C} \to \mathscr{C}$ equipped with a *strength*, *i.e.* a transformation

$$\theta^T_{\Gamma,A} : \Gamma \times T(A) \to T(\Gamma \times A)$$

natural in Γ and A, and satisfying unarity and associativity constraints. Such strong functors have already been considered in the past, see for example [8, 7, 9]. The difference here is the presence of functional types, which forces us to consider a notion dual to strengths, that we call *contravariant strengths*. A contravariant functor $N : \mathscr{C}^{op} \to \mathscr{C}$ is *strong* if there is a transformation:

$$\rho^N_{\Gamma,A} : \Gamma \times N(\Gamma \times A) \to N(A)$$

which is natural in A, dinatural in Γ and satisfies the following unarity and associativity constraints:

$$\begin{tikzcd}
1 \times N(A) \arrow[d, "1\times N(\pi_2)"'] \arrow[dr, "\pi_2"] & \\
1 \times N(1 \times A) \arrow[r, "\rho^N_{1,A}"'] & N(A)
\end{tikzcd}$$

$$\begin{tikzcd}
 & B \times (A \times N((A \times B) \times C)) \arrow[dr, "B\times(A\times N(\alpha^{-1}_{A,B,C}))"] & \\
(B \times A) \times N((A \times B) \times C)) \arrow[ur, "\alpha_{B,A,N((A\times B)\times C)}"] & & B \times (A \times N(A \times (B \times C))) \arrow[d, "B\times\rho^N_{A,B\times C}"'] \\
(A \times B) \times N((A \times B) \times C) \arrow[u, "s_{A,B}\times N((A\times B)\times C)"'] \arrow[dr, "\rho^N_{A\times B,C}"'] & & B \times N(B \times C) \arrow[dl, "\rho^N_{B,C}"] \\
 & N(C) &
\end{tikzcd}$$

Let us now take any object Γ of $\mathscr{C}$, and consider the comonad $\Gamma \times -$. It gives rise to a co-Kleisli category denoted $\mathscr{C}_\Gamma$, and corresponds to the category of morphisms (terms) in the context Γ. The main interest of strong (co/contra)-variant functors is that they can be extended to $\mathscr{C}_\Gamma$ in the following way:

Proposition 1. *Let $T : \mathscr{C} \to \mathscr{C}$ and $N : \mathscr{C}^{op} \to \mathscr{C}$ be strong functors. If we define:*

- *On objects, $T_\Gamma(A) = T(A)$. On morphisms, if $f : A \to B$ (in $\mathscr{C}_\Gamma$, hence $f : \Gamma \times A \to B$ in $\mathscr{C}$):*

$$T_\Gamma(f) = \Gamma \times T(A) \xrightarrow{\theta^T_{\Gamma,A}} T(\Gamma \times A) \xrightarrow{T(f)} T(B)$$

- *On objects, $N_\Gamma(A) = N(A)$. On morphisms, if $f : B \to A$ (in $\mathscr{C}_\Gamma$),*

$$N_\Gamma(f) = \Gamma \times N(A) \xrightarrow{\Gamma \times N(f)} \Gamma \times N(\Gamma \times B) \xrightarrow{\rho^N_{\Gamma,B}} N(B)$$

Then $T_\Gamma : \mathscr{C}_\Gamma \to \mathscr{C}_\Gamma$ and $N_\Gamma : \mathscr{C}_\Gamma^{op} \to \mathscr{C}_\Gamma$ are well-defined functors.

Moreover, it can be proved that this functor extension operation preserves the existence of initial algebras/terminal coalgebras. More precisely, we can prove the following proposition:

Proposition 2. *Let $T : \mathscr{C} \to \mathscr{C}$ be strong and Γ be an object of $\mathscr{C}$, then we have the following propositions:*

- *If T has an initial algebra in $\mathscr{C}$, then T_Γ has an initial algebra in $\mathscr{C}_\Gamma$.*
- *If T has a terminal coalgebra in $\mathscr{C}$, then T_Γ has a terminal coalgebra in $\mathscr{C}_\Gamma$.*

While this structure already allows to interpret most of the extended rules of μLJ, there are still some gaps. In particular, what guarantees that the behaviour of T is not modified when building T_Γ ? More precisely, the $(-)_\Gamma$ construction has to satisfy a certain number of equations like $(T \times T')_\Gamma = T_\Gamma \times T'_\Gamma$. These equations reduce to properties of strengths, which lead to the following definition.

Definition 1. *A category $\mathscr{C}$ has* strong types *if it is cartesian closed, has (weak, functorial) coproducts, a (weak) initial object, and is equipped with a class $\mathscr{F}$ of functors $T : \mathscr{C}^k \times (\mathscr{C}^{op})^p \to \mathscr{C}$ satisfying the following properties:*

- *$\mathscr{F}$ contains the identity, constant functors and base constructors $-_1 + -_2$, $-_1 \times -_2$ and $-_1 \Rightarrow -_2$;*
- *$\mathscr{F}$ is stable by composition : if $F, G \in \mathscr{F}$ and F, G are composable, then $FG \in \mathscr{F}$;*
- *$\mathscr{F}$ is stable by* contraction *: if $F(-_1, -_2, -_3) : \mathscr{C} \times \mathscr{C} \times \mathscr{D} \to \mathscr{C}$ is in $\mathscr{F}$, then $F(-_1, -_1, -_3) : \mathscr{C} \times \mathscr{D} \to \mathscr{C}$. Same condition with $\mathscr{C}^{op}$ instead of $\mathscr{C}$ at the left.*

Those functors in $\mathscr{F}$ that are also unary *(*i.e. *$P : \mathscr{C} \to \mathscr{C}$ or $N : \mathscr{C}^{op} \to \mathscr{C}$) are* strong. *The strengths have to satisfy the following conditions:*

- *For both covariant and contravariant strengths, the families $\theta^T_{\Gamma,A}$ and $\rho^N_{\Gamma,A}$ are also* natural *in T/N;*
- *Compatibility with identity:*

$$\theta^{-}_{\Gamma,A} = id_{\Gamma \times A}$$

- *Compatibility with constant functors:*

$$\theta^{B}_{\Gamma,A} = \pi_2$$

- *Compatibility with composition:*

$$\begin{array}{lrcl}
\textit{F and G covariant:} & \theta^{FG}_{\Gamma,A} & = & \theta^{F}_{\Gamma,G(A)};F(\theta^{G}_{\Gamma,A}) \\
\textit{F covariant and G contravariant:} & \rho^{FG}_{\Gamma,A} & = & \theta^{F}_{\Gamma,G(\Gamma\times A)};F(\rho^{G}_{\Gamma,A}) \\
\textit{F contravariant and G covariant:} & \rho^{FG}_{\Gamma,A} & = & \Gamma\times F(\theta^{G}_{\Gamma,A});\rho^{F}_{\Gamma,G(A)} \\
\textit{F and G contravariant:} & \theta^{FG}_{\Gamma,A} & = & \Gamma\times F(\rho^{G}_{\Gamma,A});\rho^{F}_{\Gamma,G(\Gamma\times A)}
\end{array}$$

- *Compatibility with contraction, with P covariant and N contravariant:*

$$\begin{array}{rcl}
\theta^{P(-,-)}_{\Gamma,A} & = & \langle\pi_1,\theta^{P(-,A)}_{\Gamma,A}\rangle;\theta^{P(\Gamma\times A,-)}_{\Gamma,A} \\
\rho^{N(-,-)}_{\Gamma,A} & = & \langle\pi_1,\rho^{N(-,\Gamma\times A)}\rangle;\rho^{N(A,-)}_{\Gamma,A}
\end{array}$$

- *Compatibility with cartesian closed structure.*

$$\begin{array}{rcl}
\rho^{-\Rightarrow C}_{\Gamma,A} & = & \Lambda(\langle\langle\pi_2;\pi_1,\pi_1\rangle,\pi_2;\pi_2\rangle;ev) \\
\theta^{C\Rightarrow -}_{\Gamma,A} & = & \Lambda(\langle\pi_2;\pi_1,\langle\pi_1,\pi_2;\pi_2\rangle;ev\rangle)
\end{array}$$

In a category with strong types, all the equations required for the expansion of functor rules hold:

Proposition 3. *For any $P,P' : \mathscr{C}\to\mathscr{C}$, $N,N' : \mathscr{C}^{op}\to\mathscr{C}$, Γ, the following equations hold:*

$$\begin{array}{rcl}
(P+P')_\Gamma & = & P_\Gamma + P'_\Gamma \\
(N+N')_\Gamma & = & N_\Gamma + N'_\Gamma \\
(P\times P')_\Gamma & = & P_\Gamma \times P'_\Gamma \\
(N\times N')_\Gamma & = & N_\Gamma \times N'_\Gamma \\
(N\Rightarrow P)_\Gamma & = & N_\Gamma \Rightarrow P_\Gamma \\
(P\Rightarrow N)_\Gamma & = & P_\Gamma \Rightarrow N_\Gamma
\end{array}$$

Moreover, if $T : \mathscr{C}\times\mathscr{D}\to\mathscr{C}$ in $\mathscr{F}$ has a parametrized initial algebra or a parametrized terminal coalgebra T^μ/T^ν :

$$\begin{array}{rcl}
(T^\mu)_\Gamma & = & (T_\Gamma)^\mu \\
(T^\nu)_\Gamma & = & (T_\Gamma)^\nu
\end{array}$$

4 μ-Closed Categories

Now that we have all the necessary background to interpret the extended rules for functors and their expansion, we can turn to the rules for fixpoints. We define *μ-closed categories* by analogy with the definition of μ-bicomplete categories [23], as those categories with strong types where the canonical interpretation of μLJ formulas as strong functors is total.

Definition 2. *Let $\mathscr{C}$ be a category with strong types. We define a partial interpretation of μLJ formulas as strong functors in the class $\mathscr{F}$ as follows:*

- $[\![0]\!] = X\mapsto 0$ *(the constant functor on the (weak) initial object);*
- $[\![1]\!] = X\mapsto 1$ *(the constant functor on the terminal object);*

- $[\![X]\!] = X \mapsto X$ *(the identity functor);*
- $[\![S \Rightarrow T]\!] = [\![S]\!] \Rightarrow [\![T]\!]$*;*
- $[\![S + T]\!] = [\![S]\!] + [\![T]\!]$*;*
- $[\![S \times T]\!] = [\![S]\!] \times [\![T]\!]$
- $[\![\mu X.T]\!]$ *is, if defined, the parametrized initial algebra of* $[\![T]\!](X, \overrightarrow{Y})$*;*
- $[\![\nu X.T]\!]$ *is, if defined, the parametrized terminal coalgebra of* $[\![T]\!](X, \overrightarrow{Y})$.

In both cases, the notation $\overrightarrow{Y}$ *expresses the fact that, since* T *can have other free type variables,* $[\![T]\!]$ *can admit other arguments than* X.

Definition 3. *Let* $\mathscr{C}$ *a category with strong types.* $\mathscr{C}$ *is* μ-closed *if the interpretation function* $[\![-]\!]$ *is total on* $\mathscr{F}$.

Theorem 1. *Any* μ*-closed category is a sound model for* μLJ *with the extended rules set.*

5 The Games Model

Let $\mathscr{I}$ denote the usual category of arenas and innocent strategies (see for example [20]), and $\mathscr{G}$ denote the category of games for fixpoints introduced in [6]. The extension of $\mathscr{I}$ and $\mathscr{G}$ to take the structure presented here into account goes rather smoothly, in several steps.

Proposition 4. $\mathscr{I}$ *has strong types, with* open functors *as the needed class of functors.*

Now, this strong types structure extends naturally to $\mathscr{G}$ by the addition of parity winning conditions. But we already know that in $\mathscr{G}$, open functors definable by the base constructors have initial algebras and terminal coalgebras, hence we get the following theorem.

Theorem 2. $\mathscr{G}$ *is* μ*-closed.*

6 Conclusion

This work presents a categorical setting in which it is possible to deal with rules for fixpoints and functors under a given context Γ. This is important, since any plausible programming language with induction and/or coinduction will allow iteration of a functional with free variables. We also show how the games model of [6] can be adapted to this extended setting, thus giving a model of the extended rules of μLJ, with games and winning total strategies.

An interesting open question is whether the definition of μ-closed category is actually redundant, *i.e.* whether the derivations of the extended rules in μLJ which are known to exist behave as needed, from the dynamical point of view. More precisely, it is possible to define strength candidates for all functors built out of the base constructors, but proving that these strength candidates satisfy the required equations turned out to be unfeasible by hand, most notably for the case of strengths for functors already generated by fixpoints (necessary for the modelling of *interleaving* inductive/coinductive types, *i.e.* embedded fixpoints within the definition of fixpoints). We note however that such verifications could be at least partially automatized, thus leading to a rather strong theorem allowing to interpret the extended rules of μLJ in any model of the basic rules.

References

[1] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *J. of Symb. Log.*, 59(2):543–574, 1994.

[2] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Proc. of 13th IEEE Ann. Symp. on Logic in Computer Science, LICS '98 (Indianapolis, IN, June 1998)*, pp. 334–344. IEEE CS Press, 1998.

[3] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Inform. and Comput.*, 163(2):409–470, 2000.

[4] A. Arnold and D. Niwinski. *Rudiments of μ-Calculus*, *Int. Series in Comput. Sci.*. Prentice Hall, 2001.

[5] D. Baelde and D. Miller. Least and greatest fixed points in linear logic. In N. Dershowitz and A. Voronkov, eds., *Proc. of 14th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2007 (Yerevan, Oct. 2007)*, v. 4790 of *Lect. Notes in Artif. Intell.*, pp. 92–106. Springer, 2007.

[6] P. Clairambault. Least and greatest fixpoints in game semantics. In L. de Alfaro, ed., *Proc. of 12th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2009 (York, March 2009)*, v. 5504 of *Lect. Notes in Comput. Sci.*, pp. 16–31. Springer, 2009.

[7] J. R. B. Cockett and D. Spencer. Strong categorical datatypes I. In R. A. G. Seely, ed., *Proc. of Int. Summer Category Theory meeting (Montréal, June 1991)*, v. 13 of *Canadian Math. Soc. Conf. Proc.*, pp. 141–169. AMS, 1992.

[8] J. R. B. Cockett and D. Spencer. Strong categorical datatypes II: a term logic for categorical programming. *Theor. Comput. Sci.*, 139(1–2):69–113, 1995.

[9] J. R. B. Cockett and T. Fukushima. About Charity. Technical report, University of Calgary, 1992.

[10] J. H. Conway. *On Numbers and Games*. AK Peters, 2001.

[11] J. De Lataillade. Second-order type isomorphisms through game semantics. *Ann. of Pure and Appl. Logic*, 151(2-3):115–150, 2008.

[12] P. J. Freyd. Algebraically complete categories. In A. Carboni et al., eds., *Proc. of Int. Conf. on Category Theory, CT '90 (Como, July 1990)*, v. 1488 of *Lect. Notes in Math.*, pp. 95–104. Springer, 1991.

[13] P. J. Freyd. Remarks on algebraically compact categories. In M. P. Fourman et al., eds., *Applications of Categories in Computer Science: Proc. of LMS Symp. (Durham, July 1991)*, v. 177 of *London Math. Soc. Lect. Note Series*, pp. 95–106. Cambridge Univ. Press, 1992.

[14] P. J. Freyd. Recursive types reduced to inductive types. In *Proc. of 5th Ann. IEEE Symp. on Logic in Computer Science, LICS '90 (Philadelphia, PA, June 1990)*, pp. 498–507. IEEE CS Press, 1990.

[15] J. Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, v. 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1989.

[16] R. Harmer. Innocent game semantics. Lecture notes, 2004.

[17] J. M. E. Hyland and C. H. L. Ong. On full abstraction for PCF: I, II, and III. *Inform. Comput.*, 163(2):285–408, 2000.

[18] A. Joyal. Remarques sur la théorie des jeux à deux personnes. *Gaz. des Sci. Math. du Québec*, 1(4):46–52, 1977.

[19] P. Lorenzen. Logik und Agon. *Atti Congr. Internat. di Filosofia*, v. 4, pp. 187–194. 1960.

[20] G. McCusker. Games and full abstraction for FPC. *Inform. and Comput.*, 160(1-2):1–61, 2000.

[21] P.-A. Melliès. Asynchronous games 4: a fully complete model of propositional linear logic. In *Proc. of 20th Ann. IEEE Symp. on Logic in Computer Science, LICS 2005 (Chicago, IL, June 2005)*, pp. 386–395. IEEE CS Press, 2005.

[22] L. Santocanale. A calculus of circular proofs and its categorical semantics. In M. Nielsen and U. Engberg, eds., *Proc. of 5th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2002 (Grenoble, Apr. 2002)*, v. 2303 of *Lect. Notes in Comput. Sci.*, pp. 357–371. Springer, 2002.

[23] L. Santocanale. μ-bicomplete categories and parity games. *Theor. Inform. and Appl.*, 36(2):195–227, 2002.

Membership Checking in Greatest Fixpoints Revisited

Martin Hofmann and Dulma Rodriguez
Institut für Informatik, Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München, Germany
mhofmann@ifi.lmu.de and rodrigue@ifi.lmu.de

Abstract

Pierce, in his book "Types and Programming Languages" (MIT Press, 2002), presents an efficient algorithm for computing membership in the greatest fixpoint of *invertible* operators in a goal-directed way. In this paper, we provide a new proof of correctness for it based on coinduction. Moreover, we extend the algorithm for computing membership in the gfp of arbitrary monotone operators and prove this extension correct in a very similar way. Finally, we instantiate the general algorithm to gain a subtyping algorithm for RAJA programs.

1 Introduction

We are interested in computing membership in the greatest fixpoint of a monotone operator on the powerset of some given set. Rather than computing the entire fixpoint by Knaster-Tarski iteration we want to depart from a given goal. This may be advantageous if the size of the underlying set or of the greatest fixpoint is large compared to the portion relevant for determining membership of a particular element. For a concrete example consider the operator $\mathcal{F}(X) = \{x \mid x+1 \mod 5 \in X\}$ on the powerset of $G = \{0,\dots,2^{100}\}$. Obviously, the largest fixpoint consists of G itself; determining this by Knaster-Tarski iteration is infeasible though. If we only want to check whether a particular element, say 23 is in the gfp we can commence with the goal $23 \in \mathsf{gfp}?$. This leads to the sequence of subgoals $4 \in \mathsf{gfp}?$, $0 \in \mathsf{gfp}?$, $1 \in \mathsf{gfp}?$, $2 \in \mathsf{gfp}?$, $3 \in \mathsf{gfp}?$, $4 \in \mathsf{gfp}?$ at which point we are done because we have discovered a loop in the sequence of subgoals that have arisen.

We are specially interested in deciding subtyping for RAJA types. The RAJA system is a refinement of an extension of Featherweight Java (FJ) [6] with attribute update (FJEU), with the goal of statically analysing the heap space consumption of object-oriented programs. The system has been first described by Hofmann and Jost in [4]. Recently, the current authors analysed algorithmic typing of RAJA programs [5]. Briefly, RAJA types are FJEU classes refined with a possibly infinite set of *views*. Subtyping for RAJA types is defined as the greatest fixpoint of a monotone operator, similarly to the definitions of subtyping for other recursive types like tree types or μ-types [7, Chapter 21].

Subtyping algorithms for recursive types have been widely studied in the past. Amadio and Cardelli gave the first subtyping algorithm for recursive types [1]. Brandt and Henglein's [2] showed the underlying coinductive nature of Amadio and Cardelli's algorithm. In [7, Chapter 21] Pierce gives an overview of many algorithms for membership checking for greatest fixed points and how they can be used to decide subtyping for recursive types.

RAJA subtyping, however, is a bit more complicated than most of the other definitions of subtyping for recursive types because in RAJA methods can have many different types. Therefore, in order to check that a RAJA type C^r is a subtype of a RAJA type D^s we need to check that for a given method m for all its method types in D^s there is a method type in C^r with some properties. This causes that the *support* of a given goal is not a set of subgoals as usual but a boolean combination of subgoals.

In this paper we will extend the efficient algorithm for membership checking for greatest fixed points described in [7, Chapter 21.6] to a more general version where the support of a given goal is a positive boolean expression. Moreover, we provide a new proof of correctness for both algorithms. We found the proof in [7, Chapter 21.6] difficult to extend and provide therefore a more abstract coinductive proof which can be easily adapted to the new algorithm.

Contents. In Section 2 we describe and prove correct an algorithm for membership checking in greatest fixed points of monotone operators closed under intersection. In Section 3 we extend the algorithm to arbitrary monotone operators. In Section 4 we instantiate the second algorithm in order to decide subtyping for the RAJA system.

2 Invertible Operators

Let G be a set. We let $\mathcal{P}(G)$ denote the powerset of G and $\mathcal{PF}(G)$ denote the set of finite subsets of G. If $\mathcal{F}:\mathcal{P}(G)\rightarrow\mathcal{P}(G)$ is a monotone operator we write $\mathsf{gfp}(\mathcal{F})$ for its greatest fixpoint. We have $\mathsf{gfp}(\mathcal{F})=\mathcal{F}(\mathsf{gfp}(\mathcal{F}))$ and whenever $X\subseteq\mathcal{F}(X)$ then $X\subseteq\mathsf{gfp}(\mathcal{F})$. The latter principle is called coinduction. We may also use the notation $\nu X.\mathcal{F}(X)$ for $\mathsf{gfp}(\mathcal{F})$.

In the following we review a goal-directed algorithm for membership checking for greatest fixed points described in [7, Chapter 21.6]. This algorithm works only for a special kind of operators, *invertible operators*, which we now characterize.

A given element $g\in G$ can be generated by a monotone operator $\mathcal{F}$ in many ways, which means that there can be more than one set $X\subseteq G$ such that $g\in\mathcal{F}(X)$. We call any such set a *generating* set for g. We focus here on the class of invertible operators, where each g has at most one minimal generating set.

Definition 2.1. *A monotone operator $\mathcal{F}$ is said to be* invertible *if, for all $g\in G$, the collection of sets*

$$G_g=\{X\subseteq G\mid g\in\mathcal{F}(X)\}$$

is either empty or contains a unique finite member that is a subset of all the others.

When F is invertible, the partial function $\mathsf{support}_{\mathcal{F}}:G\rightarrow\mathcal{PF}(G)$ is defined like this:

$$\mathsf{support}_{\mathcal{F}}(g)=\begin{cases} X & \text{if } X\in G_g \text{ and } \forall X'\in G_g\,.X\subseteq X' \\ \uparrow & \text{if } G_g=\emptyset \end{cases}$$

That is, the support of a goal g is the least generating set X for g, or undefined if g is not supported in $\mathcal{F}$.

Definition 2.2. *Let G be a set, $A\subseteq G$, $f:G\rightarrow\mathcal{PF}(G)$. A monotone operator $\mathcal{F}_{f,A}$ is defined by*

$$\begin{array}{lcl} \mathcal{F}_{f,A} & : & \mathcal{P}(G)\rightarrow\mathcal{P}(G) \\ \mathcal{F}_{f,A}(X) & = & \{g\mid g\in A\wedge f(g)\subseteq X\} \end{array}$$

Then, the support of a goal is given by the function f and it is only defined for elements $g\in A$:

$$\mathsf{support}_{\mathcal{F}_{\mathsf{f,A}}}(g)=\begin{cases} f(g) & \text{if } g\in A \\ \uparrow & \mathsf{otherwise} \end{cases}$$

The following result seems to be folklore, well-known e.g. in the field of predicate transformers. The operators $\mathcal{F}_{f,A}$ are equivalent to invertible operators and to monotone operators closed under intersection where every goal has a finite support.

Theorem 2.3. *Let $\mathcal{F}:\mathcal{P}(G)\rightarrow\mathcal{P}(G)$ be a monotone operator. The following are equivalent:*

1. *There exists f,A such that $\mathcal{F}=\mathcal{F}_{f,A}$.*

2. *For each $g\in\mathcal{F}(G)$ there exists a finite support set $S\in\mathcal{PF}(G)$ such that $g\in\mathcal{F}(S)$ and for all $X_1,X_2\in\mathcal{PF}(G)$ one has $\mathcal{F}(X_1\cap X_2)=\mathcal{F}(X_1)\cap\mathcal{F}(X_2)$.*

3. *$\mathcal{F}$ is invertible.*

Algorithm 1.

```
test        :  $G \times \mathcal{P}(G) \to \mathcal{P}(G)_\bot$

test$(g,U)$ =  if $g \in U$ then $U$
               else if $g \notin A$ then fail
               else
                   let $\{h_1,\dots,h_n\} = f(g)$ in
                   let $V_1 = \mathsf{test}(h_1, U \cup \{g\})$ in
                   let $V_2 = \mathsf{test}(h_2, V_1)$ in
                   ...
                   let $V_n = \mathsf{test}(h_n, V_{n-1})$ in
                     $V_n$
```

Figure 1: Algorithm for membership checking of greatest fixed points.

Membership checking

Figure 1 shows an algorithm for membership checking in the greatest fixed point of $\mathcal{F}_{f,A}$. The idea of this membership algorithm is to run $\mathcal{F}$ backwards: to check membership for an element g, we need to ask how g could have been generated by $\mathcal{F}$. The advantage of an invertible $\mathcal{F}$ is that there is at most one way to generate a given g. We have to be careful though, a goal g might be supported e.g. by the same goal g. If we do not detect these kind of loops, the algorithm will not terminate. Therefore we keep a set of assumptions U that is empty at the beginning and that will be incremented with every goal we handle. This way we are able to detect a loop if we check whether the current goal is a member of the set of assumptions, in which case we finish with a positive answer. The following algorithm takes a set of assumptions U as an argument and returns another set of assumptions as a result. This allows it to record the subtyping assumptions that have been generated during completed recursive calls and reuse them in later calls. For failure we use the convention: if an expression B fails, then $\mathsf{let}\ A = B\ \mathsf{in}\ C$ also fails.

This algorithm has been described and proved correct in [7, Chapter 21.6]. In [3], Costa Seco and Caires have used it as well for defining subtyping for a class-based object oriented language where classes are first class polymorphic values. We provide here a more abstract correctness proof based on coinduction.

Theorem 2.4.

1. *if G is a finite set the* $\mathsf{test}(g,U)$ *terminates.*
2. $\mathsf{test}(g,\emptyset) = V \iff g \in \nu X.\mathcal{F}_{f,A}(X)$.

Proof.

1. Termination of the algorithm follows using $|G \setminus U|$ as a ranking function.
2. Let $\mathcal{N}(U) := \nu X.\{h \mid h \in U \vee (h \in A \wedge f(h) \subseteq X)\}$. Note that $\mathcal{N}(U) = \nu X.U \cup \mathcal{F}_{F,A}(X)$. Consequently, $\mathcal{N}(\emptyset) = \nu X.\mathcal{F}_{f,A}(X)$. The goal follows then from the more general results:
   (a) $\mathsf{test}(g,U) = V \Rightarrow g \in \mathcal{N}(U)$ and $U \subseteq V \subseteq \mathcal{N}(U)$.
   (b) $\mathsf{test}(g,U) = \mathsf{fail} \Rightarrow g \notin \mathcal{N}(U)$.

which we prove simultaneously by induction on the runtime of the computation of $\mathsf{test}(g,U)$.

Case $g \in U$. Then $\mathsf{test}(g,U) = U$ by definition and $g \in \mathcal{N}(U)$ since $U \subseteq \mathcal{N}(U)$.

Case $g \notin U$ and $g \notin A$. Then $\mathsf{test}(g,U) = \mathsf{fail}$ and $g \notin \mathcal{N}(U)$ since $\mathcal{N}(U) \subseteq U \cup A$.

Case $g \notin U$ and $g \in A$. We consider the representative case $f(g) = \{h_1, h_2\}$.

Case $\mathsf{test}(h_1, U \cup \{g\}) = V_1$ and $\mathsf{test}(h_2, V_1) = V_2$.
Then by induction hypothesis we get $h_1 \in \mathcal{N}(U \cup \{g\})$ and $U \cup \{g\} \subseteq V_1 \subseteq \mathcal{N}(U \cup \{g\})$ and $h_2 \in \mathcal{N}(V_1)$ and $V_1 \subseteq V_2 \subseteq \mathcal{N}(V_1)$. From monotonicity of $\mathcal{N}(.)$ then follows $\mathcal{N}(V_1) \subseteq \mathcal{N}(\mathcal{N}(U \cup \{g\})) = \mathcal{N}(U \cup \{g\})$ easily [1], hence, we get $f(g) \subseteq \mathcal{N}(U \cup \{g\})$ (*).
Next we claim that $\mathcal{N}(U) = \mathcal{N}(U \cup \{g\})$. One direction is clear by monotonicity of $\mathcal{N}(.)$. For the other direction we use coinduction with $X_0 = \mathcal{N}(U \cup \{g\})$. To conclude $X_0 \subseteq \mathcal{N}(U)$ we thus have to prove $X_0 \subseteq U \cup \{h \mid h \in A \wedge f(g) \subseteq X_0\}$ which we now do. Pick $h \in X_0 = \mathcal{N}(U \cup \{g\})$.
From the definition of $\mathcal{N}(.)$ we get that $h \in U$ or $h = g$ or $f(g) \subseteq X_0$. The first and third case immediately yield the desired result. In the second case ($g = h$) we get $f(g) \subseteq X_0$ from (*). So we proved $\mathcal{N}(U) = \mathcal{N}(U \cup \{g\})$. Then we have $f(g) \subseteq \mathcal{N}(U)$ and $g \in A$, thus, we get the desired $g \in \mathcal{N}(U)$. Moreover, we get $U \subseteq U \cup \{g\} \subseteq \mathcal{N}(U \cup \{g\}) \subseteq \mathcal{N}(U)$.

Case $\mathsf{test}(h_1, U \cup \{g\}) = \mathsf{fail}$. Then $\mathsf{test}(g,U) = \mathsf{fail}$ and by I.H. $h_1 \notin \mathcal{N}(U \cup \{g\})$, thus, $f(g) \nsubseteq \mathcal{N}(U)$ and $g \notin \mathcal{N}(U)$.

Case $\mathsf{test}(h_1, U \cup \{g\}) = V_1$ and $\mathsf{test}(h_2, V_1) = \mathsf{fail}$. Then by induction hypothesis $U \cup \{g\} \subseteq V_1 \subseteq \mathcal{N}(U \cup \{g\})$ and $h_2 \notin \mathcal{N}(V_1)$. Moreover we have by monotonicity of $\mathcal{N}(.)$ that $\mathcal{N}(U) \subseteq \mathcal{N}(U \cup \{g\}) \subseteq \mathcal{N}(V_1)$, thus $h_2 \notin \mathcal{N}(U)$ and consequently $g \notin \mathcal{N}(U)$ as desired.

□

3 Arbitrary Monotone Operators

In this section we extend the previous algorithm to an algorithm for membership checking in the greatest fixpoint of not necessarily invertible monotone operators, where the support of a given goal is the meaning of some positive boolean expression. As mentioned in the introduction, this extension is motivated by the subtyping relation of the RAJA system. In the following we describe formally positive boolean expressions and their meaning.

Definition 3.1. Positive boolean expressions *over G are defined by the grammar*

$$e ::= \mathsf{tt} \mid \mathsf{ff} \mid g \mid e_1 \wedge e_2 \mid e_1 \vee e_2$$

where g ranges over elements of G. Let $\mathsf{PBool}(G)$ *be the set of positive boolean expressions over G.*

Positive boolean expressions denote predicates on $\mathcal{P}(G)$. In particular, g denotes $\{X \mid g \in X\}$. Formally, if $X \subseteq G$ we define the *meaning* $[\![e]\!]^X : \mathsf{bool}$ as follows:

$$\begin{array}{lcl}
[\![\mathsf{tt}]\!]^X & = & \mathsf{tt} \\
[\![\mathsf{ff}]\!]^X & = & \mathsf{ff} \\
[\![g]\!]^X & = & g \in X \\
[\![e_1 \wedge e_2]\!]^X & = & [\![e_1]\!]^X \wedge [\![e_2]\!]^X \\
[\![e_1 \vee e_2]\!]^X & = & [\![e_1]\!]^X \vee [\![e_2]\!]^X
\end{array}$$

[1] $\mathcal{N}(\mathcal{N}(U)) = \mathcal{N}(U)$ follows by monotonicity of $\mathcal{N}(.)$ and coinduction.

Example 3.2. *Let $G = \{a,b,c,d\}$ and $e = a \wedge (b \vee c)$, then $[\![e]\!]^{\{a,b\}} = \mathsf{tt}$ and $[\![e]\!]^{\{b,c\}} = \mathsf{ff}$.*

Note that $X \subseteq Y$ implies $[\![e]\!]^X \Rightarrow [\![e]\!]^Y$.

Definition 3.3. *Let $f : G \to \mathsf{PBool}(G)$ be a boolean operator. Then we obtain a monotone operator $\mathcal{F}_f$ as follows:*

$$\begin{array}{ccc} \mathcal{F}_f & : & \mathcal{P}(G) \to \mathcal{P}(G) \\ X & \mapsto & \{g \mid [\![f(g)]\!]^X = \mathsf{tt}\} \end{array}$$

Next we prove constructively that, whenever a set G is finite, we can provide a boolean operator for any monotone operator over G. We notice though that the so constructed boolean operator might be very big, hence, applying the algorithm we are about to describe would be very inefficient.

Theorem 3.4. *If G is a finite set and $\mathcal{F} : \mathcal{P}(G) \to \mathcal{P}(G)$ then there exists $f : G \to \mathsf{PBool}(G)$ such that $\mathcal{F} = \mathcal{F}_f$.*

Proof. For each (finite) subset $X = \{g_1, \ldots, g_k\} \subseteq G$ define $\bigwedge X := g_1 \wedge \ldots \wedge g_k$. We have $[\![\bigwedge X]\!]^Y = \mathsf{tt} \iff X \subseteq Y$. Given g let $X_1 \ldots X_k$ be an enumeration of the subsets X such that $g \in \mathcal{F}(X)$. We then put $f(g) = \bigwedge X_1 \ \vee \ldots \vee \ \bigwedge X_n$. Now $g \in \mathcal{F}(X) \Rightarrow X = X_i$ for some $i \ \Rightarrow [\![\bigwedge X_i]\!]^X = \mathsf{tt} \Rightarrow [\![f(g)]\!]^X = \mathsf{tt}$. Conversely $[\![f(g)]\!]^X = \mathsf{tt} \Rightarrow X_i \subseteq X$ for some $i \ \Rightarrow g \in \mathcal{F}(X_i) \Rightarrow g \in \mathcal{F}(X)$ by monotonicity. □

For invertible operators we can provide a boolean operator directly. Given $f : G \to \mathcal{P}(G)$ as in the last section and $A \subseteq G$, define $\tilde{f}$ as follows:

$$\tilde{f}(g) = \begin{cases} \mathsf{ff} & \text{if } g \notin A \\ \bigwedge f(g) & \text{if } g \in A \end{cases}$$

Then $[\![\tilde{f}(g)]\!]^X = \mathsf{tt} \iff g \in A \wedge f(g) \subseteq X$, hence, $\mathcal{F}_{\tilde{f}}(X) = \mathcal{F}_{f,A}(X)$.

Membership checking

Figure 2 shows a new algorithm for membership in the gfp of arbitrary monotone operators whenever a boolean operator $f : G \to \mathsf{PBool}(G)$ is given. *Algorithm 2* takes a set of assumptions U as an argument and returns another set of assumptions and a boolean as a result. The difference to the first algorithm is that if the meaning of the support of a goal is tt, then the new computed set of assumptions will be returned; otherwise it will be dropped. Moreover, ff branches do not lead immediately to rejection. They can lead to a positive answer if combined by “or” with a tt branch. In the following we prove correctness and termination of the algorithm. If the basic set is finite the algorithm will terminate and the result will be correct. Otherwise, even if the basic set is infinite, if the computation of the support of a goal do not lead to an infinite chain of new goals, then the algorithm will terminate as well with a correct answer.

Theorem 3.5. *Let $f : G \to \mathsf{PBool}(G)$ and test defined as above. Let $\mathcal{N}(U) = \nu X\,.\,U \cup \mathcal{F}_f(X)$.*

1. *If $\mathsf{test}(e,U) = (b,V)$ then $[\![e]\!]^{\mathcal{N}(U)} = b$ and $U \subseteq V \subseteq \mathcal{N}(U)$.*

2. *If for each g there exists a finite set S such that $f(S) \subseteq \mathsf{PBool}(S)$ and $g \in S$ then $\mathsf{test}(g,\emptyset)$ terminates.*

Proof. 2. follows using $|S \setminus U|$ as a ranking function. For 1. we induct on the runtime of $\mathsf{test}(e,U)$ and – subordinately – on the structure of e. We note that for all $U \subseteq G$ we have $U \subseteq \mathcal{N}(U)$, $\mathcal{N}(U) = \mathcal{N}(\mathcal{N}(U))$.

Algorithm 2. Let $* \in \{\wedge, \vee\}$:

```
$\mathsf{test} : \mathsf{PBool}(G) \times \mathcal{P}(G) \to \mathsf{bool} \times \mathcal{P}(G)$

$\mathsf{test}(e_1 * e_2, U) =$ let $(b_1, V_1) = \mathsf{test}(e_1, U)$ in
                             let $(b_2, V_2) = \mathsf{test}(e_2, V_1)$ in
                               $(b_1 * b_2, V_2)$
$\mathsf{test}(g, U) =$ if $g \in U$ then $(\mathsf{tt}, U)$
                        else let $(b, V) = \mathsf{test}(f(g), U \cup \{g\})$ in
                         if $b$ then $(\mathsf{tt}, V)$ else $(\mathsf{ff}, U)$
```

Figure 2: Algorithm for membership checking in the gfp of arbitrary monotone operators.

Case $e = e_1 * e_2$. Write $(b_1, V_1) = \mathsf{test}(e_1, U)$ and $(b_2, V_2) = \mathsf{test}(e_2, V_1)$.

Inductively, we have $b_1 = [\![e_1]\!]^{\mathcal{N}(U)}$ and $U \subseteq V_1 \subseteq \mathcal{N}(U)$. Therefore, $\mathcal{N}(U) \subseteq \mathcal{N}(V_1) \subseteq \mathcal{N}(\mathcal{N}(U)) = \mathcal{N}(U)$, and thus $\mathcal{N}(V_1) = \mathcal{N}(U)$. It follows that $b_2 = [\![e_2]\!]^{\mathcal{N}(U)}$ and $U \subseteq V_1 \subseteq V_2 \subseteq \mathcal{N}(U)$. The claim then follows.

Case $e = g$.

Case $g \in U$. Then $\mathsf{test}(g, U) = (\mathsf{tt}, U)$ and obviously $[\![g]\!]^{\mathcal{N}(U)} = \mathsf{tt}$ and $U \subseteq \mathcal{N}(U)$.

Case $g \notin U$. Write $(b, V) = \mathsf{test}(f(g), U \cup \{g\})$. Inductively, we have $U \cup \{g\} \subseteq V \subseteq \mathcal{N}(U \cup \{g\})$ and $b = [\![f(g)]\!]^{\mathcal{N}(U \cup \{g\})}$.

We claim that $\mathcal{N}(U) = \mathcal{N}(U \cup \{g\})$. One direction is clear by monotonicity of $\mathcal{N}(.)$. For the other direction we use coinduction with $X = \mathcal{N}(U \cup \{g\})$. To conclude $X \subseteq \mathcal{N}(U)$ we have to prove $X \subseteq U \cup \{h \mid [\![f(h)]\!]^X = \mathsf{tt}\}$ which we now do.

Pick $h \in X = \mathcal{N}(U \cup \{g\})$.

From the definition of $\mathcal{N}(.)$ we get that $h \in U$ or $h = g$ or $[\![f(h)]\!]^X = \mathsf{tt}$. The first and third case immediately yield the desired result. In the second case ($g = h$) we get $[\![f(h)]\!]^X = \mathsf{tt}$ from the induction hypothesis. So we proved $\mathcal{N}(U) = \mathcal{N}(U \cup \{g\})$. The result is now direct from the definitions.

□

Corollary 3.6. $\mathsf{test}(e, \emptyset) = (b, \_)$ *iff* $[\![e]\!]^{\mathsf{gfp}(\mathcal{F}_f)} = b$.

4 Applications

In this section we consider a special application of the last algorithm. As we already mentioned we are specially interested in computing subtyping for RAJA types. In the following we give a brief and simplified introduction to the RAJA system and show how to instantiate the generic *Algorithm* 2 to gain a RAJA subtyping algorithm.

RAJA programs are annotated FJEU programs, created with the goal of statically analysing their heap space consumption. An FJEU program $\mathscr{C}$ is a partial finite map from class names to class definitions. Classes contain attributes and methods. The RAJA type system is a refinement of the FJEU type system.

A *refined (class) type* consists of a class C and a *view* r and is written C^r. The meaning of views is given by three maps $\Diamond()$, defining potentials, A, defining views of attributes, and M, defining refined method types. More precisely, $\Diamond() : \text{Class} \times \text{View} \to \mathbb{Q}^+$ assigns each class its potential according to the employed view. Next, $\mathsf{A} : \text{Class} \times \text{View} \times \text{Field} \to \text{View}$ determines the refined types of the fields. Finally, $\mathsf{M} : \text{Class} \times \text{View} \times \text{Method} \to \mathcal{P}(\text{Views of Arguments} \to \text{View of Result})$ assigns refined types to methods. We allow polymorphism in the sense that one method may have more than one (or no) refined typing. For more details and concrete examples we refer to [4, 5, 8].

Now we describe a simplified version of subtyping for RAJA types. The simplification disregards subclasses and potentials but shows the need for going beyond invertible operators. Let RT be the set of RAJA types. We define a monotone operator $\mathcal{F} : \mathcal{P}(\mathsf{RT} \times \mathsf{RT}) \to \mathcal{P}(\mathsf{RT} \times \mathsf{RT})$ as follows:

$$\begin{array}{rcll}
\mathcal{F}(X) & = & \{\, (C^r, D^s) \;\;|\;\; & \forall \text{ attributes } a \;.\; \mathsf{A}(C^r,a) = E^p, \mathsf{A}(D^s,a) = E^q \;.\, (E^p, E^q) \in X \\
 & & & \forall \text{ methods } \; m \;.\, \forall\, (E_1^{\beta_1}, \ldots, E_j^{\beta_j} \to E_0^{\beta_0}) \in \mathsf{M}(D^s;m) \;. \\
 & & & \quad \exists\, (E_1^{\alpha_1}, \ldots, E_j^{\alpha_j} \to E_0^{\alpha_0}) \in \mathsf{M}(C^r;m) \;. \\
 & & & \qquad (E_1^{\beta_1}, E_1^{\alpha_1}) \in X, \ldots, (E_j^{\beta_j}, E_j^{\alpha_j}) \in X, (E_0^{\alpha_0}, E_0^{\beta_0}) \in X \,\}
\end{array}$$

Then $C^r <: D^s \iff (C^r, D^s) \in \nu X . \mathcal{F}(X)$. Now, in order to apply *Algorithm* 2, we define a function $f : \mathsf{RT} \times \mathsf{RT} \to \mathsf{PBool}(\mathsf{RT} \times \mathsf{RT})$ so that $\mathcal{F}(X) = \mathcal{F}_f(X)$:

$$\begin{array}{rcl}
f(C^r, D^s) & = & \bigwedge_a (E^p, E^q) \wedge \\
 & & \bigwedge_m \bigwedge_{E_1^{\beta_1}, \ldots, E_j^{\beta_j} \to E_0^{\beta_0}} \bigvee_{E_1^{\alpha_1}, \ldots, E_j^{\alpha_j} \to E_0^{\alpha_0}} (E_1^{\beta_1}, E_1^{\alpha_1}) \wedge \ldots \wedge (E_j^{\beta_j}, E_j^{\alpha_j}) \wedge (E_0^{\alpha_0}, E_0^{\beta_0})
\end{array}$$

5 Conclusions

In this paper we extended the algorithm for membership checking for greatest fixed points described in [7, Chapter 21.6] to a more general version where the support of a given goal is a positive boolean expression. For finite sets this generalization encompasses all monotone operators. Next, we provided a new coinductive correctness proof for both algorithms. Finally, we instantiated the general membership algorithm in order to compute subtyping for RAJA types in a goal-directed way.

We believe that our new algorithm can be useful for computing subtyping for other refinement systems that also provide multiple types to methods. As part of a prototype implementation of the RAJA system the algorithm has been implemented in Ocaml and we work currently in a formalization of its correctness proof in the theorem prover Coq.

Acknowledgements

We acknowledge support by the EU integrated project MOBIUS IST 15905 and by the DFG Graduiertenkolleg 1480 Programm- und Modell-Analyse (PUMA). We also thank Andreas Abel for valuable comments.

References

[1] R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Trans. on Program. Lang. and Syst.*, 15(4):575–631, 1993.

[2] M. Brandt and F. Henglein. Coinductive axiomatization of recursive type equality and subtyping. *Fundam. Inform.*, 33(4):309–338, 1998.

[3] J. Costa Seco and L. Caires. Subtyping first-class polymorphic components. In S. Sagiv, ed., *Proc. of 14th Europ. Symp. on Programming, ESOP 2005 (Edinburgh, Apr. 2005)*, v. 3444 of *Lect. Notes in Comput. Sci.*, pp. 342–356. Springer, 2005.

[4] M. Hofmann and S. Jost. Type-based amortised heap-space analysis (for an object-oriented language). In P. Sestoft, ed., *Proc. of 15th Europ. Symp. on Programming, ESOP 2006 (Vienna, March 2006)*, v. 3924 of *Lect. Notes in Comput. Sci.*, pp. 22–37. Springer, 2006.

[5] M. Hofmann and D. Rodriguez. Efficient type-checking for amortised heap-space analysis. In *Proc. of 23rd Int. Wksh. on Computer Science Logic, CSL 2009 (Coimbra, Sept. 2009)*, *Lect. Notes in Comput. Sci.*, Springer, to appear.

[6] A. Igarashi, B. C. Pierce, and P. Wadler. Featherweight Java: a minimal core calculus for Java and GJ. *ACM Trans. on Program. Lang. and Syst.*, 23(3):396–450, 2001.

[7] B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[8] Raja. `http://raja.tcs.ifi.lmu.de`.

A Note on the Relation between Inflationary Fixpoints and Least Fixpoints of Higher-Order

Stephan Kreutzer
Oxford University Computing Laboratory
Wolfson Building, Parks Road,
Oxford OX1 3QD, United Kingdom
kreutzer@comlab.ox.ac.uk

Martin Lange
Institut für Informatik,
Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München, Germany
martin.lange@ifi.lmu.de

Abstract

Least fixpoints of monotone functions are an important concept in computer science which can be generalised to inflationary fixpoints of arbitrary functions. This raises questions after the expressive power of these two concepts, in particular whether the latter can be expressed as the former in certain circumstances. We show that the inflationary fixpoint of an arbitrary function on a lattice of finite height can be expressed as the least fixpoint of a monotone function on an associated function lattice.

1 Introduction

Possibly the most important type of fixpoints in computer science are least fixpoints of monotone functions, with countless concepts and definitions being based on this principle, e.g. abstract data types, formal languages, semantics of programming constructs, static analysis algorithms, logical operators.

In mathematical logic, fixpoint inductions over definable functions on arbitrary structures have first been studied in generalised recursion theory (see [10]), following earlier work in recursion theory on inductive definitions in arithmetic. If $\varphi(X,x)$ is a first-order formula with a free first-order variable x and a free second-order variable X, which we call the fixpoint variable of φ, then φ defines on any structure $\mathfrak{A}$ with universe A a function f_φ on the powerset lattice on A with $f_\varphi(B) := \{a \in A : (A,a) \models_{[X\mapsto B]} \varphi(X,x)\}$. Of particular interest are formulas defining a monotone function as by Knaster and Tarski's theorem (see Section 2) every monotone function has a unique least fixpoint which can also be obtained by an explicit induction process. As monotonicity of a function is in general undecidable, first-order formulas that are positive in X (and therefore monotone) are usually considered only.

A similar but seemingly more general concept of fixpoints are provided by *inflationary* fixpoints, which exist for any function, even if they are non-monotone (see Section 2 for details). In the context of logic, inflationary fixpoints of definable functions have first been studied in the 1970s (see e.g. [11, 1]) and it has been realised that not every inflationary fixpoint over an arbitrary first-order formula can also be described as a least fixpoint over a formula positive in its fixpoint variable. This naturally leads to the question of which inflationary fixpoints can equivalently be written as least fixpoints of monotone functions.

Following this early work on fixpoint inductions over definable functions, logics featuring explicit fixpoint constructs have been studied in finite model theory and in temporal logics as means to describe classes of structures or the behavior of programs for instance. The main and decisive difference to the studies in generalised recursion theory was the introduction of explicit operators to form least or inflationary fixpoint of definable functions, which allows to nest fixpoint operators and use them in the scope of negations.

Initiated by Gurevich [4], logics involving fixpoint constructs have intensively been studied in finite model theory and descriptive complexity as an elegant way to describe computational problems in logical languages (see [3] for an extensive study of fixpoint logics). Again, the most important fixpoint logics considered in this context are logics extending first-order logic by operators to form the least fixpoint of formulas positive in their fixpoint variable or operators to form the inflationary fixpoints of arbitrary formulas. It turns out that combining first-order logic with the ability to nest and complement fixpoint operators is powerful enough so that every formula of inflationary fixpoint logic is equivalent to a formula using least fixpoints of formulas positive in their fixpoint variable. This was first proved in the context of finite structures by Gurevich and Shelah [5] and then generalised by Kreutzer [7] to arbitrary structures.

In the context of modal logics, fixpoints occur most prominently in the modal μ-calculus L_μ introduced by Kozen [6]. The importance of the μ-calculus stems from its fine balance between expressive power and complexity, as it is expressive enough to encompass commonly used specification logics such as LTL, CTL and CTL*. On the other hand, it model checking problem on finite structures is in NP$\cap$coNP and its satisfiability/validity problem is EXPTIME-complete. Besides its expressive power, the μ-calculus is still a *regular* logic because it can be embedded into monadic second-order logic (MSO). In fact, L_μ is the bisimulation-invariant fragment of MSO and hence is the most expressive regular logic invariant under bisimulation.

Being a regular logic comes with a range of restrictions, in particular the inability to count. Hence, specifications such as *a particular event occurs on all execution traces at the same time* or *every request is acknowledged* cannot be expressed in L_μ. To overcome the restriction to regular logics, extensions of the modal μ-calculus have been studied in the literature. Among those one can broadly distinguished between "first-order" fixpoint logics, i.e. logics where the fixpoint is still taken over definable functions from sets of vertices to sets of vertices but more general fixpoint constructs are allowed that least fixpoints over monotone functions, and "higher-order" fixpoint logics, where we retain monotone fixpoint inductions but allow fixpoints of operators over a function space. An example for the first approach is the *modal iteration calculus* (MIC), introduced in [2], the extension of modal logic by operators to form inflationary fixpoints of definable functions. An example of the latter is *fixpoint logic with chop* (FLC) introduced in [12], where the semantics of μ-calculus formulas is lifted from the powerset lattice of all *predicates* to the lattice of *predicate transformers* which are first-order functions from the original powerset lattice into itself. This concept has then been generalised to *higher-order fixpoint logic* (HFL), introduced in [14], which incorporates into the μ-calculus a simply typed λ-calculus used to describe predicate transformers, and functions of predicate transformers, and functions of functions of ..., etc.

For all these logics examples of non-regular properties definable in the logic have been exhibited, separating them from the modal μ-calculus. However, very little is known about the relationship between these logics. A simple complexity-theoretic argument shows that FLC cannot be embedded into MIC [8]: the expression complexity for FLC is EXPTIME-hard, i.e. there is a fixed formula s.t. model checking with this formula is already EXPTIME-hard [9]. On the other hand, MIC's data complexity is in P. Thus, if this particular FLC-formula was translatable into MIC then we would have EXPTIME = P which contradicts the time hierarchy theorem. It is open whether or not MIC is translatable into FLC.

This should be seen in the more general context of the question whether monotone fixpoint of higher order can be used to express non-monotone fixpoints of first order and if this cannot be achieved in general, then under which circumstances inflationary fixpoints can be expressed as monotone fixpoints of higher order. (Note that MIC uses inflationary fixpoints of functions of type $\tau \to \tau$ while FLC uses least fixpoints of functions of type $(\tau \to \tau) \to (\tau \to \tau)$.)

The purpose of this paper is to stipulate a discussion of this problem. To initiate this we present a general result on fixpoints in complete lattices show that any inflationary fixpoint on a complete lattice of finite height can be expressed as a least fixpoint of a monotone operator on a function space associated with the lattice. As a consequence, we obtain some embeddability results for modal fixpoint logics.

2 Complete Lattices and Fixpoints

2.1 Lattices

A *partial order* is a pair $(M, \leq)$ s.t. M is a set and $\leq$ is a reflexive, anti-symmetric and transitive binary relation on M. As usual, we write $<$ for the strict relation obtained from it, i.e. $< := \leq \setminus =$.

An *upper*, resp. *lower bound* for a $N \subseteq M$ is a $y \in M$ s.t. $x \leq y$, resp. $y \leq x$, for all $x \in N$. A *maximum*, resp. *minimum*, of some $N \subseteq M$ is a $y \in N$ s.t. there is no $x \in N$ with $y < x$, resp. $x < y$. A *supremum*, resp. *infimum*, of some $N \subseteq M$ is a minimum of all upper bounds, resp. maximum of all lower bounds. As usual, we write $\bigsqcup N$, resp. $\bigsqcap N$, for the supremum, resp. infimum, of N if it exists uniquely. If $N = \{x, y\}$ we also use infix relation $x \sqcup y$, resp. $x \sqcap y$.

A *lattice* is a partial order $(M, \leq)$ s.t. for every $x, y \in M$ the supremum $x \sqcup y$ and the infimum $x \sqcap y$ exists uniquely in M. It is *complete* if $\bigsqcup N$ and $\bigsqcap N$ exist uniquely in M for every $N \subseteq M$. We define $\bot = \bigsqcap M$ and $\top = \bigsqcup M$ as the bottom and top element of a complete lattice.

Function lattices Let $\mathscr{M} = (M, \leq_{\mathscr{M}})$ and $\mathscr{N} = (N, \leq_{\mathscr{N}})$ be lattices. The space of all functions from $\mathscr{M}$ to $\mathscr{N}$ is $\mathscr{M} \to \mathscr{N} := (\{f \mid f : M \to N\}, \leq)$, where

$$f \leq g \text{ iff } \forall x \in M : f(x) \leq_{\mathscr{N}} g(x)$$

Clearly, $\mathscr{M}$ need not be a lattice, not even a partial order, for the function space to be a lattice. If $\mathscr{N}$ is a lattice then so is $\mathscr{M} \to \mathscr{N}$ with

$$(f \sqcup g)(x) \;=\; f(x) \sqcup_{\mathscr{N}} g(x) \qquad (f \sqcap g)(x) \;=\; f(x) \sqcap_{\mathscr{N}} g(x)\ .$$

If $\mathscr{N}$ is complete, then so is $\mathscr{M} \to \mathscr{N}$.

2.2 Fixpoints

Let $\mathscr{M} = (M, \leq_{\mathscr{M}})$ and $\mathscr{N} = (N, \leq_{\mathscr{N}})$ be partial orders. A function $f : M \to N$ is called *monotone* if for all $x, y \in M$: if $x \leq_{\mathscr{M}} y$ then $f(x) \leq_{\mathscr{N}} f(y)$.

Let $\mathscr{M} = (M, \leq)$ be a lattice and $f : M \to M$. A *least fixpoint* of f is an element $x \in M$ s.t. $f(x) = x$ and there is no $y < x$ s.t. $f(y) = y$. Probably the most famous fixpoint theorem is Knaster-Tarski's which states unique existence of least fixpoints in case of monotone functions on complete lattices. We write μf or $\mu x.f(x)$ for *the* least fixpoint of f if it exists uniquely.

Theorem 1 ([13]). *Let $\mathscr{M} = (M, \leq)$ be a complete lattice and $f : M \to M$ monotone. Then $\mu f = \bigsqcap\{y \mid f(y) \leq y\}$.*

Another characterisation of least fixpoints of monotone functions is given by *fixpoint iteration* stating that the least fixpoint also equals the supremum of all its *approximants* $\mu^\alpha f$ for any ordinal α, defined as follows.

$$\mu^0 f := \bot \ , \quad \mu^{\alpha+1} f = f(\mu^\alpha f) \ , \quad \mu^\kappa f = \bigsqcup_{\alpha<\kappa} \mu^\alpha f$$

where κ is a limit ordinal. Then $\mu f = \bigsqcup_\alpha \mu^\alpha f$.

It is well-known and easy to show by induction that the sequence of approximants is monotonically increasing, i.e. for all ordinals α, β: if $\alpha \leq \beta$ then $\mu^\alpha f \leq_{\mathscr{M}} \mu^\beta f$. This, however, requires monotony and is not true in general for non-monotonic functions. On the other hand, the monotonous increase of the sequence is appealing for it is bound to become stable – possibly at some transfinite ordinal. Stability of course means reaching a fixpoint. If f is not monotone then one can enforce a monotonically increasing and eventually stable sequence by making it *inflationary*. The *inflationary fixpoint* of an arbitrary function $f : M \to M$ is written $\mathsf{ifp} f$ or $\mathsf{ifp} x.f(x)$ and is defined as $\bigsqcup_\alpha \mathsf{ifp}^\alpha f$ where

$$\mathsf{ifp}^0 f := \bot \ , \quad \mathsf{ifp}^{\alpha+1} f = \mathsf{ifp}^\alpha f \sqcup_{\mathscr{M}} f(\mathsf{ifp}^\alpha f) \ , \quad \mathsf{ifp}^\kappa f = \bigsqcup_{\alpha<\kappa} \mathsf{ifp}^\alpha f$$

It is not difficult to see that inflationary fixpoints are at least as expressive as least fixpoints. If f is monotone then $\mathsf{ifp} f = \mu f$. In fact, the correspondence is even stronger: $\mathsf{ifp}^\alpha f = \mu^\alpha f$ for every ordinal α. Hence, for monotone functions inflationary and least fixpoints not only coincide, they inherently are the same. This raises the question after the converse: can inflationary fixpoints be expressed in terms of least fixpoints? The next section shows that this is sometimes the case. Note that, for an arbitrary function $f : M \to M$, the function $f' : M \to M$, defined as $f'(x) = x \sqcup f(x)$ is in general not monotone and may therefore not have a (unique) least fixpoint.

In order to prove a correspondence between inflationary fixpoints and least fixpoints of higher-order in the following section we generalise the context of inflationary fixpoint iteration. Let $\mathscr{M} = (M, \leq)$ be a complete lattice, $x \in M$ and $f : M \to M$. Define $\mathsf{ifp}_x f = \bigsqcup_\alpha \mathsf{ifp}_x^\alpha f$ where

$$\mathsf{ifp}_x^0 f := x \ , \quad \mathsf{ifp}_x^{\alpha+1} f = \mathsf{ifp}_x^\alpha f \sqcup_{\mathscr{M}} f(\mathsf{ifp}_x^\alpha f) \ , \quad \mathsf{ifp}_x^\kappa f = \bigsqcup_{\alpha<\kappa} \mathsf{ifp}_x^\alpha f$$

with κ being a limit ordinal. Hence, $\mathsf{ifp}_x f$ is simply the inflationary fixpoint of f when the iteration is started in x and therefore $\mathsf{ifp} f = \mathsf{ifp}_\bot f$.

The *closure ordinal* of a function f and an element $x \in M$ is the least ordinal α s.t. $\mathsf{ifp}_x^{\alpha+1} f = \mathsf{ifp}_x^\alpha f$. It is denote $cl_x(f)$. Note that $\mathsf{ifp}_x f = \mathsf{ifp}_x^{cl_x(f)} f$. We will also write $cl(f)$ instead of $cl_\bot(f)$ where $\bot$ is the infimum of the underlying complete lattice.

3 Expressing Inflationary Fixpoints as Least Fixpoints of Higher-Order

Before we can show expressibility of inflationary fixpoints through higher-order least ones we need to prove two facts about generalised inflationary fixpoints.

Lemma 2. *Let $\mathscr{M} = (M, \leq)$ be a complete lattice, $x \in M$, and $f : M \to M$. For all ordinals $\alpha < \omega$ we have* $\mathsf{ifp}_x^{\alpha+1} f = \mathsf{ifp}_{x \sqcup f(x)}^\alpha f$.

Proof. By induction on α. The base case is $\mathsf{ifp}_x^1 = \mathsf{ifp}_x^0 f \sqcup f(\mathsf{ifp}_x^0 f) = x \sqcup f(x) = \mathsf{ifp}_{x \sqcup f(x)}^0 f$. The step case is $\mathsf{ifp}_x^{\alpha+2} f = \mathsf{ifp}_x^{\alpha+1} f \sqcup f(\mathsf{ifp}_x^{\alpha+1} f) = \mathsf{ifp}_{x \sqcup f(x)}^\alpha f \sqcup f(\mathsf{ifp}_{x \sqcup f(x)}^\alpha f) = \mathsf{ifp}_{x \sqcup f(x)}^{\alpha+1} f$. □

Lemma 3. *Let $\mathscr{M} = (M, \leq)$ be a complete lattice, $x \in M$, and $f : M \to M$. Then we have $x \sqcup \mathsf{ifp}_{x \sqcup f(x)} f \leq \mathsf{ifp}_x f$.*

Proof. Note that $x \leq \mathsf{ifp}_x f$. Thus, it suffices to show $\mathsf{ifp}_{x \sqcup f(x)} f \leq \mathsf{ifp}_x f$. We will separate this into two parts. First, we will show that for every ordinal $\alpha < \omega$ we have $\mathsf{ifp}_{x \sqcup f(x)}^\alpha f = \mathsf{ifp}_x^{\alpha+1} f$. This is done by induction on α. The base case is simple: $\mathsf{ifp}_{x \sqcup f(x)}^0 f = x \sqcup f(x) = \mathsf{ifp}_x^1 f$. In the step case we have

$$\mathsf{ifp}_{x \sqcup f(x)}^{\alpha+1} f \;=\; \mathsf{ifp}_{x \sqcup f(x)}^\alpha \sqcup f(\mathsf{ifp}_{x \sqcup f(x)}^\alpha f) \;=\; \mathsf{ifp}_x^{\alpha+1} f \sqcup f(\mathsf{ifp}_x^{\alpha+1} f) \;=\; \mathsf{ifp}_x^{\alpha+2} f$$

Thus, we have

$$\mathsf{ifp}_{x \sqcup f(x)}^\omega f \;=\; \bigsqcup_{\alpha<\omega} \mathsf{ifp}_{x \sqcup f(x)}^\alpha f \;=\; \bigsqcup_{\alpha<\omega} \mathsf{ifp}_x^{\alpha+1} f \;=\; \bigsqcup_{1 \leq \alpha<\omega} \mathsf{ifp}_x^\alpha f \;=\; \bigsqcup_{\alpha<\omega} \mathsf{ifp}_x^\alpha f \;=\; \mathsf{ifp}_x^\omega f \quad (1)$$

because $\mathsf{ifp}^0_x f = x \leq \mathsf{ifp}^1_x f$.

In the second part we show that for all ordinals $\alpha \geq \omega$ we have $\mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f = \mathsf{ifp}^{\alpha}_x f$. Again, this is done by induction on α, and the base case of $\alpha = \omega$ is done in Eq. (1). The case for successor odinals is similar to the first part of the proof.

$$\mathsf{ifp}^{\alpha+1}_{x \sqcup f(x)} f \;=\; \mathsf{ifp}^{\alpha}_{x \sqcup f(x)} \sqcup f(\mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f) \;=\; \mathsf{ifp}^{\alpha}_x f \sqcup f(\mathsf{ifp}^{\alpha}_x f) \;=\; \mathsf{ifp}^{\alpha+1}_x f$$

using the hypothesis twice. Finally, the case of limit ordinals is easy, too.

$$\mathsf{ifp}^{\kappa}_{x \sqcup f(x)} f \;=\; \bigsqcup_{\alpha<\kappa} \mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f \;=\; \bigsqcup_{\omega\leq\alpha<\kappa} \mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f \;=\; \bigsqcup_{\omega\leq\alpha<\kappa} \mathsf{ifp}^{\alpha}_x f \;=\; \mathsf{ifp}^{\kappa}_x f$$

using the hypothesis on each approximant and the fact that $\mathsf{ifp}^{\alpha}_y f \leq \mathsf{ifp}^{\omega}_y f$ for every $\alpha < \omega$ and any y.

Thus, we have $\mathsf{ifp}_{x \sqcup f(x)} f = \mathsf{ifp}_x f$ and therefore in particular $x \sqcup \mathsf{ifp}_{x \sqcup f(x)} f \leq \mathsf{ifp}_x f$ which was to be shown. □

Let $\mathscr{M} = (M, \leq_{\mathscr{M}})$ be a complete lattice and $f : M \to M$ be an arbitrary function, not necessarily monotone. Let $\mathscr{M} \to \mathscr{M} = (M \to M, \leq)$ be the complete lattice of functions from $\mathscr{M}$ to $\mathscr{M}$ with the pointwise order defined above. Define a function $F_f : (M \to M) \to (M \to M)$ as follows.

$$F_f(g) \;= \lambda x. \Big(x \sqcup g\big(x \sqcup f(x)\big)\Big)$$

Lemma 4. *Let $\mathscr{M} = (M, \leq)$ be a partial order, and $f : M \to M$ arbitrary. Then F_f is monotone w.r.t. to the partial order of the function space $\mathscr{M} \to \mathscr{M}$.*

Proof. Suppose g, g' are functions of type $M \to M$ with $g \leq g'$, i.e. $g(x) \leq_{\mathscr{M}} g'(x)$ for every $x \in M$. Then $x \sqcup (g(x \sqcup f(x))) \leq_{\mathscr{M}} x \sqcup (g'(x \sqcup f(x)))$ for every such x and therefore $F_f(g) \leq F_f(g')$. □

Hence, according to the Knaster-Tarski-Theorem (Theorem 1), F_f always possesses a least fixpoint. Next we will show that this can be used to define the inflationary fixpoint of f.

Theorem 5. *Let $\mathscr{M} = (M, \leq)$ be a complete lattice with bottom element $\bot$ and $f : M \to M$ arbitrary. If $cl(f) \leq \omega$ then $\mathsf{ifp} f = (\mu F_f)(\bot)$.*

Proof. ("$\leq$") We will prove a stronger statement: for all $x \in M$ and all $\alpha \leq \omega$ we have $\mathsf{ifp}^{\alpha}_x f \leq (\mu F_f)(x)$. In the base case of $\alpha = 0$ we have

$$\begin{aligned}\mathsf{ifp}^0_x f \;&=\; x \;\leq\; x \sqcup \Big(\mu^0 F_f\big(x \sqcup f(x)\big)\Big) \;=\; \Big(\lambda y. y \sqcup \Big(\mu^0 F_f\big(y \sqcup f(y)\big)\Big)\Big)(x) \;=\; (\mu^1 F_f)(x)\\ &\leq\; (\mu F_f)(x)\end{aligned}$$

In the step case we have

$$\begin{aligned}\mathsf{ifp}^{\alpha+1}_x f \;&=\; \mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f \;\leq\; x \sqcup \mathsf{ifp}^{\alpha}_{x \sqcup f(x)} f \;\leq\; x \sqcup (\mu F_f)\big(x \sqcup f(x)\big) \;=\Big(\lambda y. y \sqcup (\mu F_f)\big(y \sqcup f(y)\big)\Big)(x)\\ &=\; (\mu F_f)(x)\end{aligned}$$

according to Lemma 2 and the fact that μF_f is a fixpoint of the function F_f. Finally,

$$\mathsf{ifp}^{\omega}_x f \;=\; \bigsqcup_{\alpha<\omega} \mathsf{ifp}^{\alpha}_x f \;\leq\; \bigsqcup_{\alpha<\omega} (\mu F_f)(x) \;=\; (\mu F_f)(x)$$

using the hypothesis for every finite ordinal α.

("$\geq$") According to Lemma 3 we have $x \sqcup \mathsf{ifp}_{x \sqcup f(x)} f \leq \mathsf{ifp}_x f$ for all $x \in M$. Hence, by $\alpha\beta$-expansion we have

$$\big(\lambda y. y \sqcup (\lambda x.\mathsf{ifp}_x f)(y \sqcup f(y))\big) \;=\; (\lambda x. x \sqcup \mathsf{ifp}_{x \sqcup f(x)} f) \;\leq_{\mathscr{M} \to \mathscr{M}}\; (\lambda x.\mathsf{ifp}_x f)$$

which shows that $\lambda x.\mathsf{ifp}_x f$ is a pre-fixpoint of F_f. According to the Knaster-Tarski-Theorem (Thm. 1) we then have

$$\mu F_f \;\leq_{\mathscr{M} \to \mathscr{M}}\; \lambda x.\mathsf{ifp}_x f$$

which immediately yields $(\mu F_f)(x) \leq \mathsf{ifp}_x f$ for any $x \in M$, in particular $(\mu F_f)(\bot) \leq \mathsf{ifp} f$. □

4 Conclusion and Further Work

An almost immediate consequence of Theorem 5 concerns the expressive power of temporal logics extending the modal μ-calculus: the modal iteration calculus can be embedded into the first-order fragment of higher-order fixpoint logic when interpreted over finite models only. Formulas of the former can inductively be transformed into formulas of the latter. The only difficult case is that of inflationary fixpoint quantifiers which are then handled by Theorem 5. The rest is easy because both logics extend modal logic. It remains to be seen in detail whether Theorem 5 can also explain the equi-expressiveness of first-order logic with inflationary fixpoints to first-order logic with least fixpoints on finite structures.

Clearly, the result presented here does not answer all questions about the relationship between least and inflationary fixpoints. Most importantly, it remains to be seen whether Theorem 5 can be extended to lattices of arbitrary height.

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, ed., *Handbook of Mathematical Logic*, v. 90 of *Studies in Logic and the Foundations of Mathematics*, pp. 739–782. North-Holland, 1977.

[2] A. Dawar, E. Grädel, and S. Kreutzer. Inflationary fixed points in modal logic. *ACM Trans. on Comput. Log.*, 5(2):282–315, 2004.

[3] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*, 2nd ed. Springer, 1999.

[4] Y. Gurevich. Toward logic tailored for computational complexity. In M. M. Richter et al., eds., *Proc. of Logic Coll. 1983 (Aachen, July 1983), Part 2: Computation and Proof Theory*, v. 1104 of *Lect. Notes in Math.*, pp. 175–216. Springer, 1984.

[5] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Ann. of Pure and Appl. Log.*, 32:265–280, 1986.

[6] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[7] S. Kreutzer. Expressive equivalence of least and inflationary fixed-point logic. *Ann. of Pure and Appl. Log.*, 130(1–3):61–78, 2004.

[8] M. Lange. *Temporal Logics beyond Regularity*. Habilitation thesis. Ludwig-Maximilians-Univ. München, 2007. (= BRICS research report RS-07-13.)

[9] M. Lange. Three notes on the complexity of model checking fixpoint logic with chop. *Theor. Inform. and Appl.*, 41:177–190, 2007.

[10] Y. N. Moschovakis. *Elementary Induction on Abstract Structures*, v. 77 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1974.

[11] Y. N. Moschovakis. On non-monotone inductive definability. *Fundam. Math.*, 82:39–83, 1974.

[12] M. Müller-Olm. A modal fixpoint logic with chop. In C. Meinel and S. Tison, eds., *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS '99 (Trier, March 1999)*, v. 1563 of *Lect. Notes in Comput. Sci.*, pp. 510–520. Springer, 1999.

[13] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pac. J. of Math.*, 5:285–309, 1955.

[14] M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In P. Gardner and N. Yoshida, eds., *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR 2004 (London, Aug./Sept. 2004)*, v. 3170 of *Lect. Notes in Comput. Sci.*, pp. 512–528. Springer, 2004.

Coalgebraic Expressions

Robert S. R. Myers
Department of Computing, Imperial College London
180 Queen's Gate, South Kensington Campus, London SW7 2AZ, United Kingdom
rm606@doc.ic.ac.uk

Abstract

We show that certain fixpoint expressions used to describe finite Kripke polynomial coalgebras can be seen as coalgebraic modal fixpoint formulae. Both the synthesis of a coalgebra from its expression and the ability to check behavioural equivalence follow from the same tableau construction. There is an associated complete equational logic, analogous to Kleene algebra, which may now be seen as an equational presentation of a fragment of the coalgebraic μ-calculus. These expressions include the regular expressions, the free Kleene algebra with tests and fragments of CCS and Linear Temporal Logic.

1 Introduction

Bonsangue, Rutten and Silva (henceforth BRS) have recently introduced certain fixpoint expressions to describe finite Kripke polynomial coalgebras [2]. The *Kripke polynomial functors* (KPF) are those endofunctors K on Set which are composed out of the identity functor, constant functors, the product and coproduct functors, the function space functor with constant domain and the finitary powerset functor. They also specify some side conditions, provided below. The *Kripke polynomial coalgebras* (KPC) are the associated K-coalgebras i.e. functions $\gamma : X \to KX$. A KPC is *finite* if its carrier set X is finite and also non-empty, a *pointed* KPC (x,γ) is a KPC γ together with a particular state $x \in X$. Examples include deterministic automata, deterministic automata on guarded strings, stream coalgebras and labelled transition systems.

To each KPF K they associate a set of fixpoint expressions Expr_K, such that every expression $\phi \in \mathsf{Expr}_K$ induces a unique finite pointed K-coalgebra $(s_\phi, \mathsf{Aut}_\phi)$ where $\mathsf{Aut}_\phi : S_\phi \to KS_\phi$ and the set S_ϕ can be thought of as the subexpressions of ϕ. They then prove that:

Every finite pointed K-coalgebra (x,γ) has an expression $\phi \in \mathsf{Expr}_K$ with $(x,\gamma) \approx (s_\phi, \mathsf{Aut}_\phi)$

where we write $(x,\gamma) \approx (x,\gamma')$ to mean that x and x' are behaviourally equivalent or *bisimilar* in the coalgebraic sense [7]. This can be seen as a generalisation of the correspondence between finite deterministic automata and the regular expressions, known as Kleene's theorem. They have also constructed a complete equational reasoning system $\equiv_K$, parametric in K. That is:

For all $\phi, \psi \in \mathsf{Expr}_K$, $(s_\phi, \mathsf{Aut}_\phi) \approx (s_\psi, \mathsf{Aut}_\psi)$ if and only if $\phi \equiv_K \psi$ is derivable.

This is analogous to the completeness of Kleene algebra: two regular expressions denote the same regular language iff their equality can be derived. Up until now, these expressions Expr_K have been seen as process algebraic, since one naturally assigns them an operational semantics. Here we show they can be seen as formulae of the coalgebraic μ-calculus [6], which is the modal μ-calculus generalised to any coalgebraic notion of transition. For every K there is an associated tableau algorithm $\mathscr{T}_K$ which may be used to decide satisfiability or dually validity of the respective coalgebraic modal fixpoint formulae. We shall discuss the following new results:

1. Each $\phi \in \mathsf{Expr}_K$ *is* a satisfiable formula of the coalgebraic μ-calculus. The satisfying model one obtains from $\mathscr{T}_K$ is precisely the automaton Aut_ϕ.

2. The equational logic $\equiv_K$ may be understood as an interesting equational presentation of a fragment of the coalgebraic μ-calculus. Then completeness of the equational system $\equiv_K$ as proved in [1] may be seen as completeness of a fragment of the coalgebraic μ-calculus.

3. To each $\phi \in \mathsf{Expr}_K$ there is a corresponding coalgebraic μ-calculus formula ϕ', such that $(s_\phi, \mathsf{Aut}_\phi)$ and $(s_\psi, \mathsf{Aut}_\psi)$ are bisimilar iff $\phi' \leftrightarrow \psi'$ is valid. This can be decided using the tableau $\mathscr{T}_K$.

4. The regular expressions, the free Kleene algebra with tests and parts of CCS and LTL all arise as fragments. We therefore obtain natural algorithms for both their synthesis and for the testing of their behavioural equivalence in a purely generic manner.

When testing behavioural equivalence, the μ-calculus formulae we consider never contain interleaving μs and νs, simplifying things considerably. Although we do not discuss complexity here, we expect our generic decision procedure to yield e.g. PSPACE-complete algorithms for testing equivalence of regular expressions and the free Kleene algebra with tests.

2 Kripke Polynomial Functors

The Kripke polynomial functors $K : \mathsf{Set} \to \mathsf{Set}$ are inductively defined:

$$K ::= Id \mid B \mid K + K \mid K \times K \mid K^A \mid \mathscr{P}_\omega K$$

- $Id : \mathsf{Set} \to \mathsf{Set}$ is the identity functor.
- $B : \mathsf{Set} \to \mathsf{Set}$ is a constant functor such that the set B is finite. We also assume that B is a join-semilattice with a bottom element. This means that B is equipped with a binary operation $\vee : B \times B \to B$ which is associative, commutative and idempotent and also a constant $\bot \in B$ with $b \vee \bot = b$ for all $b \in B$. This is the assumption made by BRS and we shall see that it is very natural. In fact every *finite* join-semilattice with a bottom element is also a lattice because it has all joins and hence all meets.
- $+ : \mathsf{Set}^2 \to \mathsf{Set}$ is defined $X + Y = \{(1,x) : x \in X\} \cup \{(2,y) : y \in Y\} \cup \{\bot, \top\}$ and if $f : X \to X'$ and $g : Y \to Y'$ then $f + g : X + Y \to X' + Y'$ is defined in the normal way, where additionally $f + g(\bot) = \bot$ and $f + g(\top) = \top$. This abnormal definition may be understood as forcing the coproduct to be a functor on the category of join-semilattices with bottom, see below.
- $\times : \mathsf{Set}^2 \to \mathsf{Set}$ is the standard Cartesian product functor.
- $(-)^A$ is the function space functor with constant finite domain A i.e. X^A is the set of functions from A to X and if $f : X \to Y$ then $f^A : X^A \to Y^A$ is defined $f^A(\alpha) = f \circ \alpha$.
- $\mathscr{P}_\omega$ is the finitary powerset functor with $\mathscr{P}_\omega X = \{A \subseteq X : A \text{ finite}\}$ and if $f : X \to Y$ then $\mathscr{P}_\omega f : \mathscr{P}_\omega X \to \mathscr{P}_\omega Y$ is the direct image of f, restricted to finite subsets of X.

Example 1. *We consider four examples:*

1. *Let* $2 = \{0,1\}$ *denote the minimal Boolean lattice. Then* $\mathsf{DA} = 2 \times Id^A$ *is the deterministic automata functor with label set A.* DA*-coalgebras* $\gamma : X \to \mathsf{DA}X$ *are precisely deterministic automata: each* γ *may be understood as an output function* $out_\gamma : X \to 2$ *where* $out_\gamma = \pi_1 \circ \gamma$ *and transition function* $trans_\gamma : X \to X^A$ *where* $trans_\gamma = \pi_2 \circ \gamma$*.*

2. *Let* $\mathsf{Tests} = \{t_1, \ldots, t_n\}$ *be a finite set, whose elements are thought of as* tests. *We may think of the set* $BA_{\mathsf{Tests}} = \mathscr{P}\mathscr{P}\mathsf{Tests}$ *as the free Boolean lattice over* Tests. *Then* $\mathsf{AGS} = BA_{\mathsf{Tests}} \times Id^{\mathsf{Atoms} \times A}$ *is the automata on guarded strings functor with tests* Tests, *label set A and* $\mathsf{Atoms} = \mathscr{P}\mathsf{Tests}$. *The* AGS*-coalgebras are precisely the deterministic automata on guarded strings [5], with parameters* Tests *and A. Finally let* $out_\gamma^{\mathsf{Tests}} = \pi_1 \circ \gamma$ *and* $trans_\gamma^{\mathsf{Tests}} = \pi_2 \circ \gamma$.

3. *Let $Prop = \{p_1, \dots, p_n\}$ be some finite set of propositional variables and $Prop_L = \mathscr{P}Prop$ be the powerset lattice. Then we call $\mathsf{Str} = Prop_L \times Id$ the $Prop_L$-stream functor.* Str*-coalgebras are streams over the lattice $Prop_L$ and the behaviour (i.e. corresponding element of the* Str*-final coalgebra [7]) of every finite* Str*-coalgebra is the infinite word $r(s)^\omega$ where $r, s \in \mathscr{P}{Prop_L}^*$ and s is not the empty word. Let $head_\gamma = \pi_1 \circ \gamma$ and $tail_\gamma = \pi_2 \circ \gamma$.*

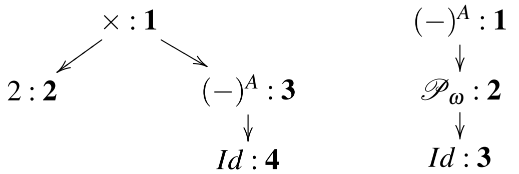
4. *Fix some finite set Act of actions, then* $\mathsf{LTS} = (\mathscr{P}_\omega Id)^A$ *is the labelled transition systems functor and* LTS*-coalgebras are labelled transition systems.*

Each KPF can also be understood as a functor on the category $\mathsf{JSL}_\bot$ of join-semilattices with bottom and semilattice morphisms that preserve the bottom. Recall each join-semilattice $(X, \vee_X)$ has a natural partial ordering $x \leq_X y \iff x \vee y = y$, the bottom is the least element with respect to this ordering. We only provide their action on objects:

- The identity functor maps join-semilattices with bottom to themselves.
- The constant functor B maps to the particular join-semilattice with bottom B, by assumption.
- The coproduct $(X+Y, \vee_+, \bot_+)$ is defined $(1,x) \vee_+ (2,y) = \top_+$, $(1,x) \vee_+ (1,x') = (1, x \vee_X x')$, $(2,y) \vee_+ (2,y') = (2, y \vee_Y y')$. The element $\bot_+ \in X+Y$ is required to be the bottom.
- $(X \times Y, \vee_{X\times Y}, \bot_{X\times Y})$ is defined $\bot_{X\times Y} = (\bot_X, \bot_Y)$ and $(x,y) \vee_{X\times Y} (x',y') = (x \vee_X x', y \vee_Y y')$.
- $(X^A, \vee_{X^A}, \bot_{X^A})$ is defined $\bot_{X^A} = \lambda a.\bot_X$ and $f \vee_{X^A} g = \lambda a.(f(a) \vee_X g(a))$.
- $(\mathscr{P}_\omega X, \vee_{\mathscr{P}_\omega X}, \bot_{\mathscr{P}_\omega X})$ is defined $\bot_{\mathscr{P}_\omega X} = \emptyset$ and $A \vee_{\mathscr{P}_\omega X} B = A \cup B$.
- Functors are closed under composition, thus every KPF can be seen as an endofunctor on $\mathsf{JSL}_\bot$.

3 Generic Syntax

We now introduce the syntax used by BRS. It differs slightly in that we have used different labels for the symbols and avoided using a type system. Syntactically we can view any KPF K as its parse tree. We think of each node of the tree as being labelled by both the respective component functor C and also a positive integer n, this being the step at which the node is visited in a depth-first left-child first search. We denote the resulting tree $Tree_K$, examples of the construction for DA and LTS are provided below.



Definition 1. *The expressions* Expr_K *associated with a KPF K are defined in terms of collection of interleaved grammars, using the structure of $Tree_K$. Let $\mathfrak{s}_1$, $\mathfrak{s}_2$ be arbitrary subtrees of $Tree_K$:*

$$\mathscr{L}_{Id:1} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \quad \mathscr{L}_{B:n} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [b] \quad (b \in B)$$
$$\mathscr{L}_{\mathfrak{s}_1 + :n\, \mathfrak{s}_2} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [e_1]\psi_1 \mid [e_2]\psi_2 \quad \mathscr{L}_{\mathfrak{s}_1 \times :n\, \mathfrak{s}_2} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi_1 \mid [\pi_2]\psi_2$$
$$(a \in A) \quad \mathscr{L}_{(\mathfrak{s}_1)^A:n} \ni \phi ::= \top \mid \phi_2 \wedge \phi_2 \mid \langle a \rangle \psi_1 \quad \mathscr{L}_{\mathscr{P}_\omega:n(\mathfrak{s}_1)} \ni \phi ::= \top \mid \phi_2 \wedge \phi_2 \mid \Diamond \psi_1$$

where $\psi_i \in \mathscr{L}_{\mathfrak{s}_i}$ for $i = 1,2$. The expressions Expr_K *are the closed and guarded members of $\mathscr{L}^\nu_{Tree_K}$:*

$$\mathscr{L}^\nu_{Tree_K} \ni \phi ::= \mathscr{L}_{Tree_K} \mid x \mid \nu x.\phi$$
$$moreover \quad \mathscr{L}_{Id:n} := \mathscr{L}^\nu_{Tree_K} \quad if\ n > 1$$

The language $\mathscr{L}^{\nu}_{Tree_K}$ is the minimal grammar containing $\mathscr{L}_{Tree_K}$, all variables x from some countably infinite set $Var = \{x_1, x_2, \dots\}$ and closed under νx-prefixing. An expression is closed and guarded if every variable x appears in the scope of some νx and variables are separated from their binders by some modal operator $\heartsuit$, in the usual way. Note that Expr_K *is inductively defined in terms of all the languages $\mathscr{L}_{\mathfrak{s}}$ where $\mathfrak{s}$ is a subtree of $Tree_K$.*

Example 2. 1. *Expressions for finite deterministic automata where $a \in A$:*

$$\mathsf{Expr}_{\mathsf{DA}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\chi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{2:2} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid [0] \mid [1] \qquad \mathscr{L}_{(Id:4)^A:3} \ni \chi ::= \top \mid \chi_1 \wedge \chi_2 \mid \langle a \rangle \phi$$

In terms of regular expressions, $\mathbf{0} = [\pi_1][0]$ should be thought of as the empty language, $\mathbf{1} = [\pi_1][1]$ as the empty word ε and $\mathbf{a}.\phi = [\pi_2]\langle a \rangle \phi$ as prefixing by the letter a. Later we show the regular expressions arise as a fragment. Using these abbreviations we obtain a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{DA}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{a}.\phi \mid x \mid \nu x.\phi$$

2. *Expressions for finite deterministic automata on guarded strings where $b \in BA_{\mathsf{Tests}} = \mathscr{P}\mathscr{P}$*$\mathsf{Tests}$, *$A \in$* $\mathsf{Atoms} = \mathscr{P}\mathsf{Tests}$ *and $a \in A$:*

$$\mathsf{Expr}_{\mathsf{AGS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\chi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{BA_{\mathsf{Tests}}:2} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid [b] \qquad \mathscr{L}_{(Id:4)^{\mathsf{Atoms} \times A}:3} \ni \chi ::= \top \mid \chi_1 \wedge \chi_2 \mid \langle (A,a) \rangle \phi$$

Because BA_{Tests} is the free Boolean algebra generated by Tests, *instead of elements $b \in BA_{\mathsf{Tests}}$ we may instead use* $\mathsf{Prop}(\mathsf{Tests})$, *the propositional formulae β with variables in* Tests. *More explicitly let $\sigma : \mathsf{Prop}(\mathsf{Tests}) \to BA_{\mathsf{Tests}}$ be the unique extension of the valuation $\sigma(t_i) = \{A \in \mathsf{Atoms} : t_i \in A\}$ for $i = 1, \dots, n$. We define $\langle \beta \rangle = [\pi_1][\beta]$ and $[\beta \to a]\phi = \bigwedge_{A \in \sigma(\beta)} [\pi_2]\langle (A,a) \rangle \phi$, so that a guarded string [5] $A_1 a_1 \dots A_{n-1} a_n A_n \in \mathsf{Atoms}(A \cdot \mathsf{Atoms})^*$ arises as the formula $[A_1 \to a_1] \dots [A_{n-1} \to a_n]\langle A_n \rangle$. This yields the single-sorted closed and guarded fragment:*

$$\mathsf{Expr}^1_{\mathsf{AGS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle \beta \rangle \mid [\beta \to a]\phi \mid x \mid \nu x.\phi$$

3. *Expressions for finite* Str*-coalgebras where $P \in Prop_L$ i.e. P is a subset of $Prop$:*

$$\mathsf{Expr}_{\mathsf{Str}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\phi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$
$$\mathscr{L}_{Prop_L:2} \ni \psi := \top \mid \psi_1 \wedge \psi_2 \mid [P]$$

We have the Next modality $\bigcirc\phi = [\pi_2]\phi$ and the Always modality $\Box\phi = \nu x.(\phi \wedge \bigcirc x)$ of LTL. Also for $P \subseteq Prop$ let $\langle P \rangle = [\pi_1][P]$. Again we have a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{Str}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle P \rangle \mid \bigcirc\phi \mid x \mid \nu x.\phi$$

4. *Expressions for finite* LTS*-coalgebras where $a \in Act$:*

$$\mathsf{Expr}_{\mathsf{LTS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle \psi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{\mathscr{P}_\omega:2(Id:3)} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid \Diamond\phi$$

For each $a \in Act$ we let $\langle \mathbf{a} \rangle \phi = \langle a \rangle \Diamond \phi$ this being the standard relational diamond. Again we have a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{LTS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle \mathbf{a} \rangle \phi \mid x \mid \nu x.\phi$$

The above languages are almost exactly the same as BRS's syntax, to obtain their languages one may bijectively relabel our symbols as follows:

$$\top \to_\theta \emptyset \qquad \phi \wedge \psi \to_\theta \phi + \psi \qquad \nu x.\phi \to_\theta \mu x.\phi$$

$$[e_1]\phi \to_\theta l[\phi] \quad [e_2]\phi \to_\theta r[\phi] \quad [\pi_1]\phi \to_\theta l(\phi) \quad [\pi_2]\phi \to_\theta r(\phi) \quad \langle a \rangle\phi \to_\theta \langle a \rangle\phi \quad \Diamond\phi \to_\theta \{\phi\}$$

The top line is rather interesting:

- The join-semilattice structure $+$ and $\emptyset$ – familiar from the regular expressions – corresponds with conjunction and $\top$. Indeed the latter also form a join-semilattice structure via propositional logical equivalence.
- Notice that to build an automaton for the regular expression $r+s$ one must build automata for *both* r and s, in this sense $+$ is a conjunction.
- To understand why $\emptyset$ corresponds with $\top$, recall that the minimal deterministic automaton that accepts the empty language is the single non-final state x where all transitions $a \in A$ loop back into x. Then ask: what is the minimal automaton which models the formula $\top$? There are two models consisting of only one state: one will accept the empty language while the other accepts its complement. In fact the former is the more natural choice because it corresponds with the bottom element of the final DA-coalgebra, which inherits its join-semilattice structure from the fact that DA is a functor on $\mathsf{JSL}_\perp$.
- Finally notice μ actually maps to ν. It is understandable that BRS decided to use μ because of the strong analogy with the Kleene star, which is best thought of as a least fixpoint and is axiomatised as such in Kleene algebra [4]. However, recall that finite deterministic automata require loops in order to accept regular languages with stars, and loops – being infinite behaviours – are represented by νs not μs.

We chose the relabellings in the second line to emphasise that the various unary operators are *modal* operators. In fact each of them is a well-known coalgebraic modal operator and the symbols we have chosen for them are standard. The multisorted languages Expr_K above and also the 'glued together' single-sorted language Expr^1_K is actually a specific application of a general construction in coalgebraic modal logic, where one builds the language and logic of composite functors out the languages and logics of their components [8].

4 Generic Final Semantics

For each K, BRS inductively define a coalgebra $\lambda_K : \mathsf{Expr}_K \to K(\mathsf{Expr}_K)$ over the expressions, which yields a finite pointed coalgebra or *automaton* $(s_\phi, \mathsf{Aut}_\phi)$ for each expression $\phi \in \mathsf{Expr}_K$. In this section we present an equivalent construction using tableau rules, whose meaning will be explained in the following section. We proceed informally since we lack the space for a detailed account.

A *sequent* Γ for $\phi \in \mathsf{Expr}_K$ is a finite subset of either Expr_K or $\mathscr{L}_\mathfrak{s}$ for some subtree $\mathfrak{s}$ of $Tree_K$. It should be thought of as the conjunction of its elements. Note it must be a finite subset of a particular component language: they may not be mixed. The following rules relate a single sequent (the premise) to one or more sequents (the conclusions).

$$(\wedge)\ \frac{\{\phi \wedge \psi\} \cup \Gamma}{\{\phi, \psi\} \cup \Gamma} \qquad (\nu)\ \frac{\{\nu x.\phi\} \cup \Gamma}{\{\phi[x := \nu x.\phi]\} \cup \Gamma} \qquad (e_i)\ \frac{\{[e_i]\phi_1, \ldots, [e_i]\phi_n\} \cup \Gamma}{\{\phi_1, \ldots, \phi_n\}} \qquad (\Diamond)\ \frac{\{\Diamond\phi_1, \ldots, \Diamond\phi_n\} \cup \Gamma}{\{\phi_1\} \quad \ldots \quad \{\phi_n\}}$$

$$(\pi)\ \frac{\{[\pi_1]\phi_1, \ldots, [\pi_1]\phi_m, [\pi_2]\psi_1, \ldots, [\pi_2]\psi_n\} \cup \Gamma}{\{\phi_1, \ldots, \phi_m\} \qquad \{\psi_1, \ldots, \psi_n\}} \qquad (\langle a_1 \ldots a_m \rangle)\ \frac{\{\langle a_1 \rangle\phi_1^1, \ldots, \langle a_1 \rangle\phi_1^{n_1}, \ldots, \langle a_m \rangle\phi_m^1, \ldots, \langle a_m \rangle\phi_m^{n_m}\} \cup \Gamma}{\{\phi_1^1, \ldots, \phi_1^{n_1}\} \quad \ldots \quad \{\phi_m^1, \ldots, \phi_m^{n_m}\}}$$

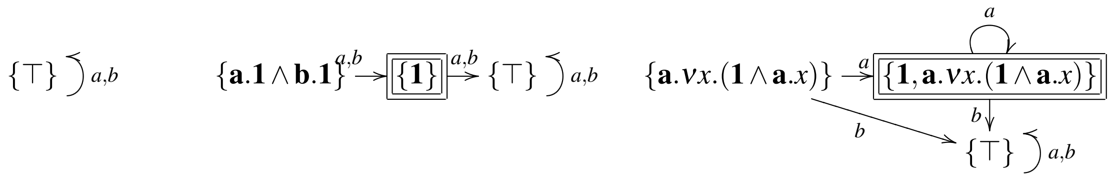
There are various conditions on how these rules are applied but we only provide them informally here. Start with the single node or sequent $\{\phi\}$ and then apply a rule whose premise can be written as $\{\phi\}$, repeating this matching process on each of the rule's conclusions. When choosing a rule to apply, the rules $(\wedge)$ and (ν) take precedence over the others. When no rule may be applied or when we come across a sequent we have already seen, we don't match rules to that sequent. The repetition of a sequent correspond with a *loop* in the automaton if the first occurrence of the sequent appears higher in the derivation tree. Roughly speaking, the process terminates because there are only finitely many possible sequents, namely the finite subsets of the subexpressions of ϕ.

Example 3. *We provide three examples of tableau:* $\top$, $\mathbf{a}.\mathbf{1}\wedge\mathbf{b}.\mathbf{1}$ *and* $\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)$ *from* $\mathsf{Expr}^1_{\mathsf{DA}}$. *Recall that* $\mathbf{1} := [\pi_1][1]$ *and* $\mathbf{a}.\phi := [\pi_2][a]\phi$ *in* $\mathsf{Expr}_{\mathsf{DA}}$.

$$\{\top\} \qquad \dfrac{\{[\pi_2]\langle a\rangle[\pi_1][1]\wedge[\pi_2]\langle b\rangle[\pi_1][1]\}}{\dfrac{\{[\pi_2]\langle a\rangle[\pi_1][1],[\pi_2]\langle b\rangle[\pi_1][1]\}}{\emptyset \qquad \dfrac{\{\langle a\rangle[\pi_1][1],\langle b\rangle[\pi_1][1]\}}{\dfrac{\{[\pi_1][1]\}}{\{[1]\}\qquad\emptyset}\,(\pi) \qquad (\{[\pi_1][1]\})}\,(\langle ab\rangle)}\,(\pi)}\,(\wedge) \qquad \dfrac{\{[\pi_2]\langle a\rangle(\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x))\}}{\emptyset \qquad \dfrac{\{\langle a\rangle(\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x))\}}{\dfrac{\{\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\dfrac{\{[\pi_1][1]\wedge[\pi_2]\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\dfrac{\{[\pi_1][1],[\pi_2]\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\{[1]\}\qquad(\{\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\})}\,(\pi)}\,(\wedge)}\,(\nu)}\,(\langle a\rangle)}\,(\pi)$$

Sequents are enclosed in brackets $(\cdot)$ *if they already appear earlier in the construction.*

Each expression ϕ yields a unique tableau which in turn completely determines the automaton $(s_\phi,\mathsf{Aut}_\phi)$. Again we will not explicitly provide the process used to convert the tableau to an automaton, although it is not complicated and is of course generic in K. Instead we provide the deterministic automata correspondents of the tableaux above, where final states are enclosed in a double box and we assume $A=\{a,b\}$:



Note that each automaton has a 'sink-node' $\{\top\}$ which corresponds with the empty-language. Notice also that the right-most automaton has a loop on $\{\mathbf{1},\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)\}$, this corresponds with the duplication of that same sequent in the tableau above and the fact that it occurred higher-up in the tableau. A duplication also occurs in the middle tableau but because the original isn't higher we don't get a loop. In fact these are automata for the regular expressions: $\emptyset$, $a+b$ and aa^*.

When BRS construct the automaton for an expression they first inductively define $\lambda_K : \mathsf{Expr}_K \to K(\mathsf{Expr}_K)$ and then show how to construct an automaton from it: this requires checking for loops to ensure termination. We have briefly outlined a method where one first constructs a tableau and then converts it to an automaton. Their process can then be understood as converting the tableau to an automaton *as the tableau is being built* i.e. it composes the two steps we have sketched. Thus the generalisation of Kleene's theorem follows:

Theorem 1. *[1] For every finite pointed KPC* (x,γ) *with* $\gamma : X\to KX$ *there exists an expression* $\phi\in\mathsf{Expr}_K$ *with* $(\{\phi\},Aut_\phi\}$ *bisimilar to* (x,γ)

Proof. Follows because the above automaton construction turns out to be equivalent to BRS's and they provide an algorithm which converts any finite pointed KPC into an expression ϕ satisfying the above property. Alternatively it is possible to obtain the result in terms of join-semilattices with bottom and the semantics presented in the next section. □

5 Generic Modal Semantics and Equational Logic

For each of our four example functors $K \in \{\mathsf{DA}, \mathsf{AGS}, \mathsf{Str}, \mathsf{LTS}\}$ and any finite coalgebra $\gamma : X \to KX$ we provide a semantics $\models^{\gamma}_{K} \subseteq X \times \mathsf{Expr}^1_K$ for the single-sorted language Expr^1_K. In fact this is the glueing together of a more general multisorted semantics. Also recall that for any join-semilattice $(B, \vee_B)$ there is a natural partial ordering $x \leq_B y \iff x \vee_B y = y$.

$$x \models^{\gamma}_{K} \top \; always \qquad x \models^{\gamma}_{K} \phi_1 \wedge \phi_2 \iff x \models^{\gamma}_{K} \phi_i \; for \; i = 1,2$$
$$x \models^{\gamma}_{K} \nu x.\phi \iff \forall n \in \omega . x \models^{\gamma}_{K} \phi[x := \nu x.\phi]^n[x := \top]$$
$$x \models^{\gamma}_{\mathsf{DA}} \mathbf{a}.\phi \iff trans_{\gamma}(x)(a) \models^{\gamma}_{\mathsf{DA}} \phi \qquad x \models^{\gamma}_{\mathsf{DA}} \mathbf{0} \iff out_{\gamma}(x) \geq_2 0 \qquad x \models^{\gamma}_{\mathsf{DA}} \mathbf{1} \iff out_{\gamma}(x) \geq_2 1$$
$$x \models^{\gamma}_{\mathsf{AGS}} [\beta \to a]\phi \iff \forall A \in \sigma(\beta) . trans^{\mathsf{Tests}}_{\gamma}(A, a) \models^{\gamma}_{\mathsf{AGS}} \phi \qquad x \models^{\gamma}_{\mathsf{AGS}} \langle \beta \rangle \iff out^{\mathsf{Tests}}_{\gamma}(x) \geq_{BA} \sigma(\beta)$$
$$x \models^{\gamma}_{\mathsf{Str}} \bigcirc \phi \iff tail_{\gamma}(x) \models^{\gamma}_{\mathsf{Str}} \phi \qquad x \models^{\gamma}_{\mathsf{Str}} \langle P \rangle \iff head_{\gamma}(x) \geq_{Prop_L} P$$
$$x \models^{\gamma}_{\mathsf{LTS}} \langle \mathbf{a} \rangle \phi \iff \exists y \in \gamma(x)(a) . y \models^{\gamma}_{\mathsf{LTS}} \phi$$

As usual we say a state x in a K-coalgebra γ *satisfies* $\phi \in \mathsf{Expr}^1_K$ if $x \models^{\gamma}_{K} \phi$ and that ϕ is *valid* if every state of every K-coalgebra satisfies ϕ. For example, in the three deterministic automata above each state Γ satisfies every $\phi \in \Gamma$. Notice that the semantics for LTS is exactly that of the modal μ-calculus with ν and $\wedge$ but without the duals μ and $\vee$. In the same way, for each K the semantics $\models^{\gamma}_{K}$ is precisely the semantics of the *coalgebraic μ-calculus* [6], which is the natural generalisation of the modal μ-calculus. More precisely, each unary and nullary modal operator $[b], [e_i], [\pi_i], \langle a \rangle, \diamond$ can be naturally assigned a coalgebraic semantics which induces the above semantics.

The tableau construction of the previous section turns out to be a slight rewriting of the tableau construction used in the coalgebraic μ-calculus.

Theorem 2. *For all $\phi \in \mathsf{Expr}_K$, BRS's automata construction is a tableau construction which builds a satisfying model of ϕ, seen as a formula of the respective coalgebraic μ-calculus*

In [2] BRS define a complete equational logic $\equiv_K \subseteq \mathsf{Expr}^1_K \times \mathsf{Expr}^1_K$[1], generic in K. By *completeness* we mean $(\{\phi\}, \mathsf{Aut}_{\phi})$ and $(\{\psi\}, \mathsf{Aut}_{\psi})$ are bisimilar iff $\phi \equiv_K \psi$ is derivable. For our example functors their equational logic takes the following form:

$$\top \wedge \phi \equiv_K \phi \qquad \phi \wedge \psi \equiv_K \psi \wedge \phi \qquad \phi \wedge (\psi \wedge \chi) \equiv_K (\phi \wedge \psi) \wedge \chi$$
$$\nu x.\phi \equiv_K \phi[x := \nu x.\phi] \qquad \phi[x := \psi] \wedge \psi \equiv_K \psi \implies (\nu x.\phi) \wedge \psi \equiv_K \psi$$
$$\top \equiv_{\mathsf{DA}} \mathbf{a}.\top \qquad \top \equiv_{\mathsf{DA}} \mathbf{0} \qquad \mathbf{0} \wedge \mathbf{1} \equiv_{\mathsf{DA}} \mathbf{1} \qquad \mathbf{a}.\phi \wedge \mathbf{a}.\psi \equiv_{\mathsf{DA}} \mathbf{a}.(\phi \wedge \psi)$$
$$\top \equiv_{\mathsf{AGS}} [\beta \to a]\top \qquad \top \equiv_{\mathsf{AGS}} [\bot \to a]\phi \qquad \top \equiv_{\mathsf{AGS}} \langle \bot \rangle \qquad \langle \beta \rangle \wedge \langle \beta' \rangle \equiv_{\mathsf{AGS}} \langle \beta \vee \beta' \rangle$$
$$[\beta \to a]\phi \wedge [\beta' \to a]\phi \equiv_{\mathsf{AGS}} [\beta \vee \beta' \to a]\phi \qquad [\beta \to a]\phi \wedge [\beta \to a]\psi \equiv_{\mathsf{AGS}} [\beta \to a](\phi \wedge \psi)$$
$$\top \equiv_{\mathsf{Str}} \bigcirc \top \qquad \top \equiv_{\mathsf{Str}} \langle \emptyset \rangle \qquad \langle P \rangle \wedge \langle P' \rangle \equiv_{\mathsf{Str}} \langle P \cup P' \rangle \qquad \bigcirc \phi \wedge \bigcirc \psi \equiv_{\mathsf{Str}} \bigcirc(\phi \wedge \psi)$$

Also $\equiv_K$ is a congruence for each unary modal operator $\mathbf{a}.$, $[\beta \to a]$ and $\bigcirc$ and satisfies the usual axioms of equational logic, plus uniform renaming of variables and their binders. Interestingly, the equations involving νx are precisely those used by Kozen in [3], to prove the completeness of the aconjunctive fragment of the modal μ-calculus. There he writes them in their dual form, using $\vee$ and μ. Aside from the semilattice equations for $\wedge$ and $\top$ and renaming of variables, the other equations may be be understood as the rank-1 modal formulae [9] which only involve $\wedge$ and $\top$. There are no equations for $\mathscr{P}_{\omega}$ involving modal operators precisely because $\diamond$ doesn't preserve conjunctions.

6 Generic Decidability and a Translation

We mentioned in the introduction that for every expression $\phi \in \mathsf{Expr}_K$ there is an associated expression ϕ' such that: $\phi \equiv_K \psi$ if and only if $\phi' \to \psi'$ is valid i.e. every state in a coalgebra that satisfies ϕ' also

[1] Strictly speaking they do it for the multisorted version of the syntax but it follows for the single-sorted version

satisfies ψ' and vice-versa. In fact for DA, AGS and Str we have $\phi' = \phi$: no change is needed. The only problem is when K contains $\mathscr{P}_\omega$. Intuitively, to capture relations up to bisimulation one needs Hennessy-Milner logic i.e. conjunctions involving $\Box$s as well as $\Diamond$s. One can overcome this problem by changing the syntax slightly and using $\nabla\{\phi_1,\dots,\phi_n\}$, rather than conjunction and $\Diamond$ in $\mathscr{L}_{\mathscr{P}_\omega:n}$. Modulo these changes:

Theorem 3. *For each $\phi,\psi \in \mathsf{Expr}_K$ one has $\phi \equiv_K \psi$ if and only if $\phi \leftrightarrow \psi$ is valid*

Proof. Just as one associates a regular language to a regular expression one can also associate a set of formulae $Conj_K(\phi) \subseteq \mathscr{L}_{Tree_K}$, to each expression ϕ. Then one shows $\phi \equiv_K \psi \iff Conj_K(\phi) = Conj_K(\psi)$ and that for all ϕ a state in a K-coalgebra satisfies ϕ iff it satisfies every $\chi \in Conj_K$. Briefly, $Conj_{\mathsf{DA}}$, $Conj_{\mathsf{AGS}}$, $Conj_{\mathsf{LTS}}$ assign expressions regular languages, regular languages of guarded strings and formulae of Hennessy-Milner logic, respectively. Moreover the join-semilattice with bottom structure of K – which lifts to the final K-coalgebra – is crucial to the proof. □

Corollary 1 ([6])**.** *Decidability of behavioural equivalence follows from the decidability of validity of the coalgebraic μ-calculus.*

Example 4. *The regular expressions for $a \in A$ are defined:*

$$RegExp \ni r ::= \emptyset \mid \varepsilon \mid a \mid r_1 + r_2 \mid r_1 r_2 \mid r^*$$

We define the translation $\tau : RegExp \to \mathsf{Expr}^1_{\mathsf{DA}}$ by: $\tau(\emptyset) = \mathbf{0}$, $\tau(\varepsilon) = \mathbf{1}$, $\tau(a) = \mathbf{a}.\mathbf{1}$, $\tau(r_1 + r_2) = \tau(r_1) \wedge \tau(r_2)$, $\tau(r_1 r_2) = \tau(r_1)[\mathbf{1} := \tau(r_2)]$ and finally $\tau(r^*) = \nu x.(\mathbf{1} \wedge \tau(r)[\mathbf{1} := x])$. Then $\mathsf{Aut}_{\tau(r)}$ is the automaton which accepts r's regular language. Moreover any state in a finite DA-coalgebra which satisfies $\tau(r)$ also accepts each word from this language, although it may accept additional words too. As an exercise one can check that a state satisfies $\tau(aa^*) = \mathbf{a}.\nu x.(\mathbf{1} \wedge \mathbf{a}.x)$ iff it satisfies $\tau(a^*a) = \nu x.(\mathbf{a}.\mathbf{1} \wedge \mathbf{a}.x)$.

The translated formula $\tau(r) \in \mathsf{Expr}^1_{\mathsf{DA}}$ can be exponentially larger than r due to compositions e.g. $\tau((a+b)^n)$ is a binary tree of depth n. However one may coinductively define composition as follows. Let $comp : \mathsf{Expr}_K \times \mathsf{Expr}_K \to \mathsf{Expr}_K$ be $comp(\nu x.\phi, \psi) = comp(\phi[x := \nu x.\phi], \psi)$, $comp(\phi_1 \wedge \phi_2, \psi) = comp(\phi_1) \wedge comp(\phi_2, \psi)$, $comp(\mathbf{a}.\phi, \psi) = \mathbf{a}.comp(\phi, \psi)$, $comp(\top, \psi) = comp(\mathbf{0}, \psi) = \top$ and finally $comp(\mathbf{1}, \psi) = \psi$. Then $\tau((a+b)^n)$ can be represented as $comp(\tau(a+b), comp(\tau(a+b), \dots)$, which avoids the exponential blow up.

References

[1] M. M. Bonsangue, J. J. M. M. Rutten, and A. Silva. Algebras for Kripke polynomial coalgebras. In *Proc. of 24th Ann. IEEE Symp. on Logic in Comput. Sci., LICS 2009 (Los Angeles, Aug. 2009)*, IEEE CS Press, to appear.

[2] M. M. Bonsangue, J. J. M. M. Rutten, and A. Silva. A Kleene theorem for polynomial coalgebras. In L. de Alfaro, ed., *Proc. of 12th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2009 (York, March 2009)*, v. 5504 of *Lect. Notes in Comput. Sci.*, pp. 122–136. Springer, 2009.

[3] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[4] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inform. and Comput.*, 110(2):366–390, 1994.

[5] D. Kozen. On the coalgebraic theory of Kleene algebra with tests. Technical report, Dept. of Computer Sci., Cornell University, 2008. `http://hdl.handle.net/1813/10173`

[6] C. Kupke, C. Cirstea, and D. Pattinson. Complexity of the coalgebraic mu-calculus. In *Proc. of 23rd Int. Wksh. on Computer Science Logic, CSL 2009 (Coimbra, Sept. 2009)*, *Lect. Notes in Comput. Sci.*, Springer, to appear.

[7] J. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[8] L. Schröder and D. Pattinson. Modular algorithms for heterogeneous modal logics. In L. Arge et al., eds., *Proc. of 34th Int. Coll. on Automata, Languages and Programming, ICALP 2007 (Wrocław, July 2007)*, v. 4596 of *Lect. Notes in Comput. Sci.*, pp. 459–471. Springer, 2007.

[9] L. Schröder and D. Pattinson. PSPACE bounds for rank-1 modal logics. *ACM Trans. on Comput. Log.*, 10(2), article 13, 2009.

On Characteristic Formulae for Event-Recording Automata

Omer Landry Nguena Timo
LaBRI, Université Bordeaux I & CNRS
351 cours de la Libération,
F-33405 Talence Cedex, France
omer-landry.nguena-timo@labri.fr

Pierre-Alain Reynier
LIF, Université de Provence & CNRS
39 rue Joliot-Curie,
F-13453 Marseille Cedex 13, France
pierre-alain.reynier@lif.univ-mrs.fr

Abstract

A standard bridge between automata theory and logic is provided by the notion of characteristic formula. This paper investigates this problem for the class of event-recording automata. An attempt to express in Event-recording logic (ERL) characteristic formula for timed simulation and bisimulation can be found in Sorea's thesis, but appears to be erroneous. We introduce an extension of the logic ERL, called WT_{μ}. We prove it is strictly more expressive than ERL, and that its model-checking problem against event-recording automata is EXPTIME-complete. We provide constructions for characterizing event-recording automata up to timed bisimilarity, and timed similarity. Finally, combining these two results we obtain decision procedures for checking timed similarity and timed bisimilarity for event-recording automata and we study the complexity issues.

1 Introduction

In the untimed setting, automata and logics are central tools for the formal verification of reactive systems. While the system is usually modelled as an automaton, the specification may be described both as a formula of a logic or as an automaton. In the first case the correctness of the system reduces to a model checking problem, whereas in the second case it requires to compare the two automata, and different relations can be envisaged, such as bisimulation or language inclusion. A standard bridge between automata theory and logic is provided by the notion of *characteristic formula* [7, 14]. A characteristic formula is a formula in a temporal logic that completely characterizes the behaviour of an automaton modulo some chosen relation. For the class of timed automata [3], a solution has first been proposed in [8], providing formulae in greatest only fixpoint logic L_{ν}. Then, these results have been improved in [1], yielding linear constructions.

Event-recording automata (ERA) [4] and timed automata [3] are timed extension of finite automata through addition of a finite set of real-valued *clocks*. They have been put forward to model continuous-time real-time systems. Event-recording Automata is a restricted class of timed automata. Whereas transitions in (untimed) finite automata are labelled with actions, every transition in ERA and timed automata is labelled with a triplet made of a constraint on clocks, an action and a set of clocks to be reset when the transition is taken. In both timed models the time elapses continuously in states and the values of clocks do change accordingly. A transition is firable when the *clock constraint* in it is satisfied by the current values of clocks. Timed automata neither restrict clocks and actions in models, nor the set of clocks to be reset when transitions are taken. ERA considers a bijective mapping between the set of clocks and the set of actions; and when a transition is taken, only the unique clock associated to the action of the transition is reset. In the opposite of timed automata, ERA are closed under boolean operations [3]. It has thus attracted attention to characterize its expressive power in terms of some timed logic [11, 6], using linear-time logics. This paper investigates the problem of identifying a branching-time logic devoted to event-based specifications that allows to construct characteristic formulae for ERA. Sorea introduced such a logic, named Event-Recording Logic (ERL), which extends the fixpoint mu-calculus by allowing the use of event-clocks. However, the construction proposed in her PhD thesis [13] for bisimulation is erroneous, and we will see that ERL cannot express timed bisimilarity for ERA.

After recalling standard definitions in Section 2, we consider in Section 3 the fixpoint timed logic WT_μ [10], to express the characteristic formulae. The definition of this logic is closer from the definition of L_ν as it separates quantification over discrete successors and time successors. We prove that it is strictly more expressive than ERL, and that its model-checking problem over ERA is EXPTIME-complete. Finally, we provide formulae constructions in WT_μ for timed (bi)similarity together with complexity issues in Section 4. Then we present a bug in the ERL-based construction proposed in [13]. Due to lack of space, omitted proofs can be found in [9].

2 Preliminaries

Let Σ be a finite alphabet, Σ^* is the set of finite words over Σ. The sets $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{Q}_{\geq 0}$, $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ are respectively the sets of natural, rational, non-negative rational, real and non-negative real numbers. We consider as time domain $\mathbb{T}$ the set $\mathbb{Q}_{\geq 0}$ or the set $\mathbb{R}_{\geq 0}$. We consider a finite set $\mathscr{X}$ of variables, called *clocks*. A *clock valuation* over $\mathscr{X}$ is a mapping $v : \mathscr{X} \to \mathbb{T}$ that assigns to each clock a time value. The set of all clock valuations over X is denoted $\mathbb{T}^{\mathscr{X}}$. Let $t \in \mathbb{T}$, the valuation $v+t$ is defined by $(v+t)(x) = v(x)+t$, $\forall x \in \mathscr{X}$. For a subset r of $\mathscr{X}$, we denote by $v[r \leftarrow 0]$ the valuation such that for each $x \in r$, $(v[r \leftarrow 0])(x) = 0$ and for each $x \in \mathscr{X} \setminus r$, $(v[r \leftarrow 0])(x) = v(x)$. Finally, $\mathbf{0}$ denotes the valuation mapping every clock on 0.

Given a set of clocks $\mathscr{X}$, we introduce the sets of clock constraints over $\mathscr{X}$ denoted by $\mathscr{C}(\mathscr{X})$, and defined by the grammar "$g ::= x \sim c \mid g \wedge g$" where $x \in \mathscr{X}$, $c \in \mathbb{Q}_{\geq 0}$, $\sim \in \{<, \leq, =, \geq, >\}$ and we define the always true constraint $\mathtt{tt} := \bigwedge_{x \in X} x \geq 0$. The set of *guards* over $\mathscr{X}$ is defined by the grammar " $\xi ::= g \mid \xi \vee \xi \mid \neg \xi$ " where g is a clock constraint over X. We write $v \models \xi$ (or $v \in [\![\xi]\!]$) when the clock valuation v satisfies ξ. The guard $\neg\xi$ stands for the negation of ξ: $v \in [\![\neg\xi]\!]$ iff $v \notin [\![\xi]\!]$.

2.1 Timed Transition Systems and Timed Behavioral Relations

Timed transition systems describe systems which combine discrete and continuous evolutions. They are used to define the behavior of timed systems [3, 4]. A *timed transition system (*TTS*)* over the alphabet Σ is a transition system $\mathscr{S} = \langle Q, q_0, \Sigma, \rightarrow \rangle$, where Q is the set of states, $q_0 \in Q$ is the initial state, and the transition relation $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{T}) \times Q$ consists of continuous transitions $q \xrightarrow{d} q'$ (with $d \in \mathbb{T}$), and discrete transitions $q \xrightarrow{a} q'$ (with $a \in \Sigma$). Moreover, we require the following standard properties for TTS: TIME-DETERMINISM (if $q \xrightarrow{d} q'$ and $q \xrightarrow{d} q''$ with $d \in \mathbb{R}_{\geq 0}$, then $q' = q''$), 0-DELAY ($q \xrightarrow{0} q$), ADDITIVITY (if $q \xrightarrow{d} q'$ and $q' \xrightarrow{d'} q''$ with d, $d' \in \mathbb{R}_{\geq 0}$, then $q \xrightarrow{d+d'} q''$), and CONTINUITY (if $q \xrightarrow{d} q'$, then for every d' and d'' in $\mathbb{R}_{\geq 0}$ such that $d = d' + d''$, there exists q'' such that $q \xrightarrow{d'} q'' \xrightarrow{d''} q'$). With these properties, a *run* of S is defined as a finite sequence of moves $\rho = q_0 \xrightarrow{d_0} q'_0 \xrightarrow{a_0} q_1 \xrightarrow{d_1} q'_1 \xrightarrow{a_1} q_2 \ldots \xrightarrow{a_n} q_{n+1}$ where discrete and continuous transitions alternate. To such a run corresponds the timed word $w = (a_i, \tau_i)_{0 \leq i \leq n}$ over Σ, where a_i occurs at time $\tau_i = \sum_{j=0}^{i} d_j$; and we say that w belong to the language of $\mathscr{S}$ denoted by $\mathscr{L}(\mathscr{S})$.

Definitions of timed simulation and timed bisimulation are given for TTS and they will be used for ERA. Consider two TTS $\mathscr{S}_1 = \langle Q_1, q_0^1, \Sigma, \rightarrow_1 \rangle$ and $\mathscr{S}_2 = \langle Q_2, q_0^2, \Sigma, \rightarrow_2 \rangle$. A *timed simulation between $\mathscr{S}_1$ and $\mathscr{S}_2$* is a relation $\mathscr{R} \subseteq Q_1 \times Q_2$ such that whenever $q_1 \mathscr{R} q_2$ and $\alpha \in \Sigma \cup \mathbb{T}$, then:

- If $q_1 \xrightarrow{\alpha} q'_1$ then there exists $q'_2 \in Q_2$ such that $q_2 \xrightarrow{\alpha} q'_2$ and $q'_1 \mathscr{R} q'_2$.

A *timed bisimulation between $\mathscr{S}_1$ and $\mathscr{S}_2$* is a relation $\mathscr{R} \subseteq Q_1 \times Q_2$ such that whenever $q_1 \mathscr{R} q_2$ and $\alpha \in \Sigma \cup \mathbb{T}$, then:

- If $q_1 \xrightarrow{\alpha} q'_1$ then there exists $q'_2 \in Q_2$ such that $q_2 \xrightarrow{\alpha} q'_2$ and $q'_1 \mathscr{R} q'_2$.

- If $q_2 \xrightarrow{\alpha} q_2'$ then there exists $q_1' \in Q_1$ such that $q_1 \xrightarrow{\alpha} q_1'$ and $q_1'\mathscr{R}q_2'$.

We write $q_1 \prec q_2$ (resp. $q_1 \sim q_2$) iff there exists a timed simulation (resp. a timed bisimulation) $\mathscr{R}$ with $q_1\mathscr{R}q_2$. Finally, we say that *a* TTS $\mathscr{S}_2$ *simulates a* TTS $\mathscr{S}_1$ (resp. $\mathscr{S}_1$ *and* $\mathscr{S}_2$ *are bisimilar*) whenever there exists a timed simulation (resp. a timed bisimulation) between $\mathscr{S}_1$ and $\mathscr{S}_2$ such that the pair (q_0^1, q_0^2) of their initial states belongs to the relation $\mathscr{R}$, and then we write $\mathscr{S}_1 \prec \mathscr{S}_2$ (resp. $\mathscr{S}_1 \sim \mathscr{S}_2$).

2.2 Event-Recording Automata

We consider the class of Event-Recording Automata (ERA), introduced in [4]. In this context, each clock refers to a specific action. Then, we associate clocks with letters of an alphabet. Given an alphabet Σ, we then denote by $\mathscr{X}_\Sigma$ the set of clocks $\{x_a \mid a \in \Sigma\}$. Intuitively, in any configuration, the value of the clock x_a represents the delay elapsed since the last occurrence of the action a (or since the beginning of the run if no action a occurred yet).

An *event-recording automaton(*ERA*) [4]* over the alphabet Σ is a tuple $\mathscr{A} = \langle L, \ell_0, \Sigma, T\rangle$ where, L is a finite set of locations, $\ell_0 \in L$ is the initial location, and $T \subseteq L \times \mathscr{C}(\mathscr{X}_\Sigma) \times \Sigma \times L$ is a finite set of transitions. An ERA is deterministic if $[\![g' \wedge g'']\!] = \emptyset$ whenever (ℓ, g', a, ℓ') and (ℓ, g'', a, ℓ'').

The semantics of an event-recording automaton $\mathscr{A}$, is defined in the terms of a timed transition system. Intuitively, it manipulates exactly one clock per action, which allows to measure time elapsed since the last occurrence of this action. The formal definition is given by: given an ERA $\mathscr{A} = \langle L, \ell_0, \Sigma, T\rangle$, its semantics is given by the TTS $\mathscr{S}_\mathscr{A}$ defined by $\mathscr{S}_\mathscr{A} = \langle Q, q_0, \Sigma, \rightarrow\rangle$ where $Q = L \times \mathbb{T}^{\mathscr{X}_\Sigma}$, $q_0 = (\ell_0, \mathbf{0})$, and $\rightarrow$ consists of continuous and discrete moves:

Delay steps: $\forall d \in \mathbb{T}$, we have $(\ell, \nu) \xrightarrow{d} (\ell, \nu + d)$,

Discrete steps: $\forall a \in \Sigma$, we have $(\ell, \nu) \xrightarrow{a} (\ell', \nu')$ iff there exists a transition $t = (\ell, g, a, \ell') \in T$ such that $\nu \models g$ and $\nu' = \nu[x_a := 0]$.

The language of an ERA $\mathscr{A}$, denoted $\mathscr{L}(\mathscr{A})$, is the language $\mathscr{L}(\mathscr{S}_\mathscr{A})$ of its TTS $\mathscr{S}_\mathscr{A}$. A basic problem on ERA consists in testing the emptiness of its language. As $\mathscr{S}_\mathscr{A}$ is infinite, a standard solution is based on a finite time abstract bisimulation called the region construction [4]. We assume the reader is familiar with the *region construction* of [3] for timed automata. Given an integer K, we denote by $\mathscr{R}_K(\mathscr{A})$ the region automaton w.r.t. constant K. Recall that the number of clock regions for ERA on alphabet Σ and maximal constant K is in $2^{O(|\Sigma| \log K|\Sigma|)}$ (see [4]). A standard solution to the emptiness testing considers region automata w.r.t maximal constant that occurs in ERAs.

Let $\mathscr{A}$ and $\mathscr{B}$ be two ERA. We say that $\mathscr{A}$ *simulates* $\mathscr{B}$ and we write $\mathscr{A} \prec \mathscr{B}$, (resp. $\mathscr{A}$ *and* $\mathscr{B}$ *are bisimilar* and we write $\mathscr{A} \sim \mathscr{B}$) whenever there exists a timed simulation (resp. a timed bisimulation) between $\mathscr{S}_\mathscr{A}$ and $\mathscr{S}_\mathscr{B}$. It is standard that: if $\mathscr{A} \prec \mathscr{B}$, then $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{B})$; and, if $\mathscr{B}$ is deterministic and $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{B})$, then $\mathscr{A} \prec \mathscr{B}$.

Let $\mathscr{A}$ be an ERA. We say that a sentence φ is a characteristic formula for $\mathscr{A}$ if and only if, according to the behavioural relation considered, the following equivalence holds:

[Simulation:] $\forall \mathscr{B} \in \text{ERA}, \mathscr{A} \prec \mathscr{B} \iff \mathscr{B} \models \varphi$

[Bisimulation:] $\forall \mathscr{B} \in \text{ERA}, \mathscr{A} \sim \mathscr{B} \iff \mathscr{B} \models \varphi$.

Let us introduce some notations. Given an ERA $\mathscr{A} = \langle L, \ell_0, \Sigma, T\rangle$, a location $\ell \in L$ and a letter $a \in \Sigma$, we denote by $Out(\ell, a) = \{t = (\ell, g, a, \ell') \in T\}$, the set of a-labelled transitions leaving ℓ and we denote by $F(\ell, a) = \{\ell' \mid \exists (\ell, g, a, \ell') \in Out(\ell, a)\}$, the set of locations reached by an a from location ℓ. We also define the guard $En(\ell, a) = \bigvee\{g \mid \exists(\ell, g, a, \ell') \in Out(\ell, a)\}$, the disjunction of clock constraints of a-labelled transitions leaving ℓ.

3 A μ-calculus for Event-Recording Automata

We present here a weak timed μ-calculus for ERA that has been introduced in [10]. Its definition distinguishes between delay successors and discrete successors, as it is done in the logic L_ν for instance. We show that it is strictly more expressive than the logic ERL. We will show in the next section that it allows to express timed (bi)similarity for ERA while ERL does not.

3.1 The Logic WT$_\mu$

Let Σ be a finite alphabet and *Var* be a finite set of variables. A formula φ of WT$_\mu$ is generated using the following grammar: $\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid \langle g \rangle \varphi \mid [a]\varphi \mid [g]\varphi \mid \mu X.\varphi \mid \nu X.\varphi$ where $g \in \mathscr{C}(\mathscr{X}_\Sigma)$, $a \in \Sigma$ and $X \in Var$.

As for the logic ERL, the semantics is defined for TTS associated with ERA. We use auxiliary assignment functions, and the notions of free (bound) variable, sentence...

For a given ERA $\mathscr{A} = \langle L, \ell_0, \Sigma, T \rangle$ with associated TTS $\mathscr{S}_\mathscr{A} = \langle Q, q_0, \Sigma, \rightarrow \rangle$, a given formula $\varphi \in$ WT$_\mu$, and an assignment function $\mathscr{V} : Var \rightarrow \mathscr{P}(Q)$, we define the set of states satisfying the formula, denoted $[\![\varphi]\!]^\mathscr{A}_\mathscr{V}$, inductively as follows:

- $[\![\mathtt{tt}]\!]^\mathscr{A}_\mathscr{V} := Q$
- $[\![\mathtt{ff}]\!]^\mathscr{A}_\mathscr{V} := \emptyset$
- $[\![X]\!]^\mathscr{A}_\mathscr{V} := \mathscr{V}(X)$
- $[\![\varphi_1 \wedge \varphi_2]\!]^\mathscr{A}_\mathscr{V} := [\![\varphi_1]\!]^\mathscr{A}_\mathscr{V} \cap [\![\varphi_2]\!]^\mathscr{A}_\mathscr{V}$
- $[\![\varphi_1 \vee \varphi_2]\!]^\mathscr{A}_\mathscr{V} := [\![\varphi_1]\!]^\mathscr{A}_\mathscr{V} \cup [\![\varphi_2]\!]^\mathscr{A}_\mathscr{V}$
- $[\![\langle a \rangle \varphi]\!]^\mathscr{A}_\mathscr{V} := \{(\ell, v) \in Q \mid \exists (\ell, g, a, \ell') \in T \text{ s.t. } v \models g \text{ and } (\ell', v') \in [\![\varphi]\!]^\mathscr{A}_\mathscr{V}, \text{ where } v' = v[x_a := 0]\}$
- $[\![\langle g \rangle \varphi]\!]^\mathscr{A}_\mathscr{V} := \{(\ell, v) \in Q \mid \exists\, d \in \mathbb{T} \text{ s.t. } v + d \models g \text{ and } (\ell, v + d) \in [\![\varphi]\!]^\mathscr{A}_\mathscr{V}\}$
- $[\![[a]\varphi]\!]^\mathscr{A}_\mathscr{V} := \{(\ell, v) \in Q \mid \forall (\ell, g, a, \ell') \in T, v \models g \Rightarrow (\ell', v') \in [\![\varphi]\!]^\mathscr{A}_\mathscr{V}, \text{ where } v' = v[x_a := 0]\}$
- $[\![[g]\varphi]\!]^\mathscr{A}_\mathscr{V} := \{(\ell, v) \in Q \mid \forall\, d \in \mathbb{T}, v + d \models g \Rightarrow (\ell, v + d) \in [\![\varphi]\!]^\mathscr{A}_\mathscr{V}\}$
- $[\![\mu X.\varphi]\!]^\mathscr{A}_\mathscr{V} := \cap \{Q' \subseteq Q \mid [\![\varphi]\!]^\mathscr{A}_{\mathscr{V}[X:=Q']} \subseteq Q'\}$
- $[\![\nu X.\varphi]\!]^\mathscr{A}_\mathscr{V} := \cup \{Q' \subseteq Q \mid Q' \subseteq [\![\varphi]\!]^\mathscr{A}_{\mathscr{V}[X:=Q']}\}$

An ERA $\mathscr{A} = \langle L_\mathscr{A}, \ell_0^\mathscr{A}, \Sigma, T_\mathscr{A} \rangle$ is a model of a sentence φ, and we write $\mathscr{A} \models \varphi$ if $(\ell_0, \mathbf{0}) \in [\![\varphi]\!]^\mathscr{A}$. Note that the valuation in the subscript of $[\![]\!]$ is removed for sentences.

Let ξ, g_1, g_2 be three constraints such that $[\![\xi]\!] = [\![g_1]\!] \cup [\![g_2]\!]$. One [10] can show that $\langle \xi \rangle \varphi$ is equivalent to $\langle g_1 \rangle \varphi \vee \langle g_2 \rangle \varphi$ and $[\xi]\varphi$ is equivalent to $[g_1]\varphi \wedge [g_2]\varphi$. In consequence we can extend the syntax of WT$_\mu$ by allowing guards to occurs in the modalities $\langle\rangle$ and [].

Remark ***(On greatest fixpoints)*** To express characteristic formulae, we shall see later that we need greatest fixpoints on systems of inequations. In this case, we will use a slightly different presentation. Given a finite set *Var* of variables, we will associate to each variable X a formula $\mathscr{D}(X)$ over the variables *Var*. $\mathscr{D}$ is then called a declaration, and the semantics associated with this definition is the largest solution of the system of inequations $X \subseteq \mathscr{D}(X)$ for any $X \in Var$. It can be proved (see [5]) that this presentation is equivalent. To specify the declaration used, we will add it as subscript of the satisfaction relation $\models$, writing $\mathscr{A}, q \models_\mathscr{D} X$.

3.2 Expressiveness and Model-Checking results

Relation with L_ν The logic L_ν over the finite set of clocks $\mathscr{X}$, the set of identifiers *Var*, and the set of events Σ is defined as the set of formulas generated by the following grammar[1]:
"$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid x\, \underline{in}\, \varphi \mid x \bowtie c \mid \langle a\rangle\varphi \mid [a]\varphi \mid \langle\delta\rangle\varphi \mid [\delta]\varphi \mid X \mid \nu X.\varphi(X)$", where $a \in \Sigma$, $x \in \mathscr{X}$ is a clock variable, $c \in \mathbb{Q}_{\geq 0}$, X is a variable, and $\bowtie \in \{\leq, \geq, <, >\}$.

The logic L_ν allows for the recursive definition of formulas by including a set *Var* of variables. L_ν allows only the greatest fixpoint operator. A formula is interpreted over timed automata. Here, we adapt the interpretation on an ERA $\mathscr{A}$ with associated TTS $\mathscr{S}_{\mathscr{A}} = \langle Q, q_0, \Sigma, \rightarrow\rangle$. Formulas are interpreted over *states* of the form $(\ell, v) \in Q$ where ℓ is a location of $\mathscr{A}$, v is a valuation of clocks in $\mathscr{X}_\Sigma$. We only present the semantics for the non standard operators $x \bowtie c, \langle\delta\rangle, [\delta]$, and $x\ \underline{in}\ \varphi$:

- $[\![x_a \bowtie c]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid v(x_a) \bowtie c\}$
- $[\![[\delta]\varphi]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid \forall d \in \mathbb{T}, (\ell, v+d) \in [\![\varphi]\!]^{\mathscr{A}}_{\mathscr{V}}\}$
- $[\![\langle\delta\rangle\varphi]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid \exists d \in \mathbb{T} \text{s.t. } (\ell, v+d) \in [\![\varphi]\!]^{\mathscr{A}}_{\mathscr{V}}\}$
- $[\![x_a\ \underline{in}\ \varphi]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid (\ell, v[x_a := 0]) \in [\![\varphi]\!]^{\mathscr{A}}_{\mathscr{V}}\}$

For ERA, the fragment of WT_μ without the least fixpoint operator is a fragment of L_ν [8]. This inclusion follows from the fact that the modal operators $[g]\varphi$, $\langle g\rangle\varphi$, $[a]\varphi$ and $\langle a\rangle\varphi$ of WT_μ are respectively equivalent to $[\delta](\neg g \vee \varphi)$[2], $\langle\delta\rangle(g \wedge \varphi)$, $[a](x_a\ \underline{in}\ \varphi)$ and $\langle a\rangle(x_a\ \underline{in}\ \varphi)$ of L_ν. As L_ν is a fragment of T_μ without the least fixpoint operator, we get that WT_μ is a fragment of T_μ, what justifies its name.

Relation with ERL We compare WT_μ with ERL. The syntax of ERL [12] is similar to the syntax of WT_μ, except that the modal operators for ERL are only of the form $\langle g, a\rangle$ or $[g, a]$. Their semantics is as follows:

- $[\![\langle g, a\rangle\varphi]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid \exists\, d \in \mathbb{T},\ \exists(\ell, g, a, \ell') \in T \text{ s.t. } v+d \models g \text{ and } (\ell', v+d[x_a := 0]) \in [\![\varphi]\!]^{\mathscr{A}}_{\mathscr{V}}\}$
- $[\![[g, a]\varphi]\!]^{\mathscr{A}}_{\mathscr{V}} := \{(\ell, v) \in Q \mid \forall\, d \in \mathbb{T}, \forall(\ell, g, a, \ell') \in T, v+d \models g \Rightarrow (\ell', v+d[x_a := 0]) \in [\![\varphi]\!]^{\mathscr{A}}_{\mathscr{V}}\}$

Theorem 1. WT_μ *is strictly more expressive than* ERL.

The inclusion of ERL in WT_μ is trivial (replace any operator $[g, a]$, resp. $\langle g, a\rangle$, by the two operators $[g][a]$, resp. $\langle g\rangle\langle a\rangle$). To show that WT_μ is strictly more expressive than the logic ERL, one may consider the formula $[0 \leq x_a \leq 1]\langle a\rangle$; this formula requires the existence of *some* discrete move with the event a in *all* the time instants at which the value of x_a is between 0 and 1; such an alternation of quantification cannot be expressed in ERL. An alternative proof can be found in [9].

Model-Checking Given an ERA $\mathscr{A}$ and a WT_μ sentence φ, the model-checking problem of $\mathscr{A}$ against φ consists in determining whether the relation $\mathscr{A} \models \varphi$ holds or not.

Theorem 2. *The model-checking problem for* ERA *against* WT_μ *sentences is EXPTIME-complete.*

EXPTIME membership can be deduced from the EXPTIME membership of the same problem for timed automata against L_ν [2]. More precisely, for an ERA $\mathscr{A}$ and a WT_μ formula φ, one can solve the problem in time $O((|\mathscr{R}_K(\mathscr{A})| \times |\varphi|)^{n+1})$, where K is the maximal constant in $\mathscr{A}$ and φ, and n is the number of alternations of greatest and least fixpoints quantifiers in φ. EXPTIME hardness follows from the EXPTIME hardness of the model-checking of ERA against ERL [13], as WT_μ extends ERL.

[1] This grammar is different, but equivalent to the one in [8]

[2] Note that the negation of a clock constraint is a disjunction of clock constraints, *i.e.* a guard.

4 Characteristic Formulae Constructions

In the sequel, we consider an ERA $\mathscr{A} = \langle L_{\mathscr{A}}, \ell_0^{\mathscr{A}}, \Sigma, T_{\mathscr{A}} \rangle$ over the alphabet Σ. Let $\ell \in L_{\mathscr{A}}$ and $a \in \Sigma$, we first introduce an operation, denoted $Split(\ell,a)$, related to the determinization of ERA. $Split(\ell,a)$ is a finite set of constraints $\{g_1,\dots,g_n\} \subseteq \mathscr{C}(\mathscr{X}_\Sigma)$ such that: it partitions $En(\ell,a)$ meaning that $[\![\bigvee_i g_i]\!] = [\![En(\ell,a)]\!]$ and $\forall i \neq j, [\![g_i]\!] \cap [\![g_j]\!] = \emptyset$; and secondly, its elements "match" the clock constraints of a-labelled transitions leaving ℓ manning that $\forall i \in \{1,\dots,n\}, \forall (\ell,g,a,\ell') \in T_{\mathscr{A}}, [\![g_i]\!] \subseteq [\![g]\!]$ or $[\![g_i]\!] \cap [\![g]\!] = \emptyset$. We do not investigate here how such an operator can be defined as it is not the purpose of this work. It can for instance be defined using the region construction [3], and then be optimized using some merging operations on zones. It is worth noticing that in the worst case, the size of $Split(\ell,a)$ may be $|Out(\ell,a)| \times 2^{O(|\Sigma|\log K|\Sigma|)}$, with K the largest integer constant of $\mathscr{A}$ (due to the region construction). However, if the ERA $\mathscr{A}$ is deterministic, then its size is linear in the size of $Out(\ell,a)$. Indeed, the determinism implies that the clock constraints of a-labelled transitions leaving ℓ are disjoint.

4.1 Characteristic Formulae for Timed Bisimulation

A characteristic formula characterising a location of an ERA up to timed bisimilarity should offer a description of: all the actions from the alphabet that are enabled in the location; which node is entered by taking a given transition, together with the reset associated with it; and the fact that arbitrary delays are allowed in the location.

We define a declaration $\mathscr{D}_{\sim\mathscr{A}}$ associating a formula to each location ℓ of $\mathscr{A}$, and consider the greatest solution of this system of fixpoint equations.

$$\Phi^{\sim\mathscr{A}}(\ell) \overset{\mathscr{D}_{\sim\mathscr{A}}}{=} \begin{cases} \bigwedge\limits_{a\in\Sigma} \bigwedge\limits_{(\ell,g,a,\ell')\in T_{\mathscr{A}}} [g]\langle a\rangle\, \Phi^{\sim\mathscr{A}}(\ell') \quad\wedge\quad [\mathtt{tt}]\Phi^{\sim\mathscr{A}}(\ell) & (\mathscr{C}_1) \\ \wedge & \\ \bigwedge\limits_{a\in\Sigma} \bigwedge\limits_{g\in Split(\ell,a)} [g][a] \bigvee\limits_{(\ell,g',a,\ell')\in T_{\mathscr{A}} \mid [\![g]\!]\subseteq[\![g']\!]} \Phi^{\sim\mathscr{A}}(\ell') \quad\wedge\quad \bigwedge\limits_{a\in\Sigma} [\neg En(\ell,a)][a]\mathtt{ff} & (\mathscr{C}_2) \end{cases}$$

We give some intuition on its definition. Let $\mathscr{B}$ be an ERA and analyze how the definition of $\Phi^{\sim\mathscr{A}}(\ell)$ constrains a location m of $\mathscr{B}$ that satisfies $\Phi^{\sim\mathscr{A}}(\ell)$. Assume that the current state in $\mathscr{S}_{\mathscr{A}}$ is (ℓ,v) and the current state in $\mathscr{S}_{\mathscr{B}}$ is (m,v).

The part $\mathscr{C}_1$ expresses the simulation constraints ($\mathscr{A} \prec \mathscr{B}$). The left-hand side of $\mathscr{C}_1$ is the sub-formula $\bigwedge_{a\in\Sigma} \bigwedge_{(\ell,g,a,\ell')\in T_{\mathscr{A}}} [g]\langle a\rangle\, \Phi^{\sim\mathscr{A}}(\ell')$ which requires that any discrete transition from (ℓ,v) also exists from (m,v); or more precisely, for any transition in $\mathscr{A}$ from (ℓ,v) and *for all delays* after which it is firable, *there exists* a corresponding transition from (m,v) leading to a related (bisimilar) state. The right hand-side of $\mathscr{C}_1$, $[\mathtt{tt}]\Phi^{\sim\mathscr{A}}(\ell)$, handles the case of delay transitions. Note that it would be easy to handle invariants in ERA. The part $\mathscr{C}_2$ requires any discrete transition from (m,v) to be related to some discrete transition from (ℓ,v); it also requires the target state of any discrete transition from (m,v) to be related to the target state of some discrete transition from (ℓ,v). The right-hand side of $\mathscr{C}_2$, $\bigwedge_{a\in\Sigma}[\neg En(\ell,a)][a]\mathtt{ff}$ states that a-transitions can happen from (m,v) only in the time instants at which a-transitions can happen from (ℓ,v). The left-hand side of $\mathscr{C}_2$,$\bigwedge_{a\in\Sigma}\bigwedge_{g\in Split(\ell,a)}[g][a]\bigvee_{(\ell,g',a,\ell')\in T_{\mathscr{A}} \mid [\![g]\!]\subseteq[\![g']\!]} \Phi^{\sim\mathscr{A}}(\ell')$ uses the decomposition $Split(\ell,a)$ of the guard $En(\ell,a)$ to express that any a-transition firable from (m,v) corresponds to some firable a-transition of (ℓ,v). In case of non determinism, the target state of an a-transition from (m,v) is non deterministically related to the target state of some a-transition from (ℓ,v); this choice is done according to the constraint satisfied by the valuation v. Note that the second property of the operator *Split* ensures the completeness of this construction.

Let us comment the size of the formulas. Due to the use of the operator *Split*, these formulae are in the worst case of size $|\mathscr{A}| \times 2^{O(|\Sigma|\log K|\Sigma|)}$, with K the largest integer constant of $\mathscr{A}$, whereas if $\mathscr{A}$

is deterministic, then their size is linear in the size of $\mathscr{A}$. We believe that this exponential blow-up is not avoidable, and detail why formulae of [1], which have a linear size, cannot be used directly in our context. In the second part of the formulae ($\mathscr{C}_2$), they indeed compare, after the discrete firing, the clock valuation with the guards of $\mathscr{A}$. As for ERA, when a discrete transition labelled by a is fired the clock x_a is reset, one can not recover the value of this clock x_a before the firing. We solve this problem by splitting the set $En(\ell,a)$ to determine which transitions of $\mathscr{A}$ were firable. Moreover, note that this exponential blow-up has no consequences on the theoretical time complexity of timed bisimilarity checking, as linear formulae would lead to the same complexity.

The following result states the correctness of the previous construction.

Theorem 3. *Let $\mathscr{A}$ and $\mathscr{B}$ be two* ERA *over Σ and consider ℓ and m two locations of $\mathscr{A}$ and $\mathscr{B}$ respectively. Then for any valuation $v \in \mathbb{T}^\Sigma$, we have : $(\ell,v) \sim (m,v) \iff \mathscr{B},(m,v) \models_{\mathscr{D}_{\sim\mathscr{A}}} \Phi^{\sim\mathscr{A}}(\ell)$ In particular, we have: $\mathscr{A} \sim \mathscr{B} \iff \mathscr{B} \models_{\mathscr{D}_{\sim\mathscr{A}}} \Phi^{\sim\mathscr{A}}(\ell_0^{\mathscr{A}})$.*

We only present a sketch of proof. It proceeds by double implication. The direct implication is proved by using the co-induction principle.in showing that, considering the assignment function $\mathscr{V}$ over the variables $\Phi^{\sim\mathscr{A}}(\ell)$ defined by $\mathscr{V}(\Phi^{\sim\mathscr{A}}(\ell)) = \{(m,v) \in Q_{\mathscr{B}} \mid (\ell,v) \sim (m,v)\}$ for any $\ell \in L_{\mathscr{A}}$, we have: $\forall \ell \in L_{\mathscr{A}}, [\![\Phi^{\sim\mathscr{A}}(\ell)]\!]^{\mathscr{B}}_{\mathscr{V}} \subseteq [\![\mathscr{D}_{\sim\mathscr{A}}(\Phi^{\sim\mathscr{A}}(\ell))]\!]^{\mathscr{B}}_{\mathscr{V}}$. This follows from an examination of the different conjuncts of $\Phi^{\sim\mathscr{A}}(\ell)$. Conversely, we consider the relation $\mathscr{R} \subseteq Q_{\mathscr{A}} \times Q_{\mathscr{B}}$ defined as $\mathscr{R} = \{((\ell,v),(m,v)) \mid \mathscr{B},(m,v) \models_{\mathscr{D}_{\sim\mathscr{A}}} \Phi^{\sim\mathscr{A}}(\ell)\}$ and show that it is a timed bisimulation. Intuitively conjunct $\mathscr{C}_1$ is used to prove that $\mathscr{R}$ is a timed simulation between $\mathscr{A}$ and $\mathscr{B}$, and $\mathscr{C}_2$ is used to prove that $\mathscr{R}^{-1}$ is a timed simulation between $\mathscr{B}$ and $\mathscr{A}$. □

Using our constructions, one can decide timed bisimilarity of two ERA $\mathscr{A}$ and $\mathscr{B}$ over Σ in time $|\mathscr{A}| \times |\mathscr{B}| \times 2^{O(|\Sigma| \log K |\Sigma|)}$ (K denotes the largest constant of $\mathscr{A}$ and $\mathscr{B}$). Using the previous theorem, this problem reduces to the model checking problem of $\mathscr{B}$ against formula $\Phi^{\sim\mathscr{A}}(\ell_0^{\mathscr{A}})$ under the declaration $\mathscr{D}_{\sim\mathscr{A}}$. Note that $\Phi^{\sim\mathscr{A}}$ contains only greatest fixpoints and thus is alternation-free. From the model-checking results, the time complexity of this problem is in $O(|\mathscr{R}_K(\mathscr{B})| \times |\Phi^{\sim\mathscr{A}}|)$.

The result follows from the size of $\mathscr{R}_K(\mathscr{B})$ and previous remarks on the size of the formulae $\Phi^{\sim\mathscr{A}}$.

4.2 Characteristic Formulae for Timed Simulation

We define a declaration $\mathscr{D}_{\succ\mathscr{A}}$ associating a formula to each location ℓ of $\mathscr{A}$, and consider the greatest solution of this system of fixpoint equations.

$$\Phi^{\succ\mathscr{A}}(\ell) \overset{\mathscr{D}_{\succeq\mathscr{A}}}{=} \bigwedge_{a\in\Sigma} \bigwedge_{(\ell,g,a,\ell')\in T} [g]\langle a\rangle\, \Phi^{\succ\mathscr{A}}(\ell') \quad \wedge \quad [\mathtt{tt}]\, \Phi^{\succ\mathscr{A}}(\ell) \qquad (\mathscr{C}_1')$$

This construction leads to formulae of *size linear* in the size of $\mathscr{A}$. Observe that $\mathscr{C}_1'$ is just $\mathscr{C}_1$ in the formula for timed bisimulation. The following result states the correctness of the previous construction.

Theorem 4. *Let $\mathscr{A}$ and $\mathscr{B}$ be two* ERA *over Σ and consider ℓ and m two locations of $\mathscr{A}$ and $\mathscr{B}$ respectively. Then for any valuation $v \in \mathbb{T}^\Sigma$, we have : $(\ell,v) \prec (m,v) \iff \mathscr{B},(m,v) \models_{\mathscr{D}_{\succ\mathscr{A}}} \Phi^{\succ\mathscr{A}}(\ell)$ In particular, we have: $\mathscr{A} \prec \mathscr{B} \iff \mathscr{B} \models_{\mathscr{D}_{\succ\mathscr{A}}} \Phi^{\succ\mathscr{A}}(\ell_0^{\mathscr{A}})$*

The proof is similar to that of Theorem 3. As for bisimilarity, one can decide timed similarity of two ERA $\mathscr{A}$ and $\mathscr{B}$ over Σ in time $|\mathscr{A}| \times |\mathscr{B}| \times 2^{O(|\Sigma| \log K |\Sigma|)}$ (K denotes the largest constant of $\mathscr{A}$ and $\mathscr{B}$). Moreover, using the determinization procedure for ERA, this procedure can also be used to decide in EXPTIME the language inclusion between two ERA $\mathscr{A}$ and $\mathscr{B}$ (first determinize $\mathscr{B}$, and then check timed simulation). Note that the problem of language inclusion is PSPACE-complete [4], thus this procedure is not optimal. However, the known algorithm matching the lower bound consists in guessing a path in the region automaton. A zone-based version of this procedure may thus be an interesting alternative.

4.3 Reporting a Bug in [13]

In [13], the author addresses the problem of constructing characteristic bisimulation formulae for ERA using ERL formulae with greatest fixpoints. In Section 3, we established that the formula $[0 \leq x_a \leq 1]\langle a \rangle \mathtt{tt}$ is not equivalent to any ERL formula. In general, WT_μ formulae having a sequence of the form $[g]\langle a \rangle \varphi$ [3][4] are not equivalent to some ERL formula. In the above subsection, characteristic formulae for timed bisimulation and timed simulation involve such kind of sequences. This is intuitively the reason why the construction in [13] is erroneous. More generally, using the same idea, we prove in [9]:

Theorem 5. *The logic* ERL *can not express neither timed bisimilarity nor timed similarity for* ERA.

We only give here a sketch of the proof. We consider an ERA $\mathscr{A}$ composed of two locations and a single edge labelled by a, with the constraint $0 \leq x_a \leq 1$. The proof proceeds by contradiction and assumes the existence of an ERL formula φ characterizing $\mathscr{A}$ up to timed bisimilarity. As we can suppose that φ is guarded (see [12]), it is possible to unfold the fixpoints of φ, and restrict the unfolding to depth 2 (because of the structure of $\mathscr{A}$). Then, the formula contains no more fixpoints, and can be rewritten in conjunctive normal form ($\bigwedge_i \varphi_i$). We finally build an ERA $\mathscr{B}$ with two locations, as $\mathscr{A}$, that has strictly less behaviours than $\mathscr{A}$, thus is not bisimilar to $\mathscr{A}$, but which satisfies φ. Therefore we pick for each φ_i whose outermost modality is of the form $\langle g, a \rangle$ a rational number r in g, and add a transition in $\mathscr{B}$ with constraint $x_a = r$. We can then verify that all formulae φ_i are satisfied by $\mathscr{B}$. □

5 Conclusion

We focused on the construction of characteristic formulae for ERA up to timed (bi)similarity with respect to WT_μ. We also reported a bug in an early construction with the setting of ERL.

Compared to existing results of [1] for timed automata which can also be applied to ERA using natural translations, we obtain procedures in the same class of complexity (EXPTIME), but our time complexities are more precise. For instance, for a fixed alphabet Σ and if constants are encoded in unary, then timed (bi)simulation can be checked in polynomial time! Moreover, our algorithm for model checking WT_μ against ERA should also be more efficient than going through L_ν and timed automata as it involves only one copy of the event-clocks. Finally, we obtain a non-optimal procedure for inclusion checking between ERA, which we believe could lead to good results in practice.

As future work, we plan to study how the good decidability results of the satisfiability problem for ERL transfer to WT_μ. Such decidability results could be useful for the supervisory control of real-time systems with non controlability assumptions. Ongoing work in that direction are promising.

References

[1] L. Aceto, A. Ingólfsdóttir, M. L. Pedersen, and J. Poulsen. Characteristic formulae for timed automata. *Theor. Inform. and Appl.*, 34(6):565–584, 2000.

[2] L. Aceto and F. Laroussinie. Is your model-checker on time? *J. of Log. and Algebr. Program.*, 52–53:7–51, 2002.

[3] R. Alur and D. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[4] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211(1–2):253–273, 1999.

[3] The action a should be firable in all the timing context at which g is satisfied.

[4] When modelling real-time reactive systems, a could represent an uncontrollable event from the environment.

[5] H. Bekic. Definable operation in general algebras, and the theory of automata and flowcharts. In C. B. Jones, ed., *Programming Languages and Their Definition*, v. 177 of *Lect. Notes in Comput. Sci.*, pp. 30–55. Springer, 1984.

[6] D. D'Souza. A logical characterisation of event clock automata. *Int. J. of Found. of Comput. Sci.*, 14(4):625–640, 2003.

[7] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Inform. and Control*, 68(1-3):125–145, 1986.

[8] F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic—and back. In J. Wiedermann, P. Hájek, eds., *Proc. of 20th Int. Symp. on Math. Found. of Comput. Sci., MFCS '95 (Prague, Aug./Sept. 1995)*, v. 969 of *Lect. Notes in Comput. Sci.*, pp. 529–539. Springer, 1995.

[9] O. Nguena and P.-A. Reynier. On characteristic formulae for event-recording automata. Research report HAL-00383203, HAL, CNRS, 2009.

[10] O. L. Nguena Timo. The logic WTμ. Technical report RR-1460-09, LaBRI, 2009.

[11] J.-F. Raskin and P.-Y. Schobbens. The logic of event clocks—decidability, complexity and expressiveness. *J. of Autom., Lang. and Combinat.*, 4(3):247–286, 1999.

[12] M. Sorea. A decidable fixpoint logic for time-outs. In L. Brim et al., eds., *Proc. of 13th Int. Conf. on Concurrency Theory, CONCUR 2002 (Brno, Aug. 2002)*, v. 2421 of *Lect. Notes in Comput. Sci.*, pp. 255–271. Springer, 2002.

[13] M. Sorea. *Verification of Real-Time Systems through Lazy Approximations*. PhD thesis, Univ. Ulm, 2004.

[14] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Inform. and Comput.*, 110(1):149–163, 1994.

Coinductive Predicates as Final Coalgebras

Milad Niqui* and Jan Rutten
Centrum Wiskunde & Informatica,
P. O. Box 94079, 1090 GB Amsterdam, The Netherlands
m.niqui@cwi.nl and jan.rutten@cwi.nl

Abstract

We show that coinductive predicates expressing behavioural properties of infinite objects can be themselves expressed as final coalgebras in a category of relations. The well-known case of bisimulation will simply be a special case of such final predicates. We will show how some useful pointwise and mixed properties of streams can be modelled in this way.

1 Introduction

Bisimulation is a widely used tool for proving the equivalent behaviour of infinite objects such as input/output systems and labelled transition systems. Although its original formulation was in the theory of processes and automata, later it was shown that maximal bisimulation — or *bisimilarity*— is tantamount to equality on the elements of final coalgebra. This leads to the *coinduction proof principle*: in order to prove that two elements of final coalgebra are equal, find a bisimulation between them. Furthermore, it was shown that for a certain class of functors, given any two coalgebras the set of all of bisimulation relations between them forms a complete lattice [14, Corollary 5.6]. It follows that bisimilarity is a post fixed point in the sense of Knaster-Tarski semantics.

Many have already observed that apart from equality there are other interesting binary relations on infinite objects which lead to other type of coinduction principles. Such principles are usually presented using Knaster-Tarski semantics. In this work we try to study this situation coalgebraically. Our starting point will be the well-known observation that bisimilarity itself is a final coalgebra for a different functor either in the same category (in the case of categorical models of dependent type theory [9, 8]) or in a different category (in the case of endofunctors on **Set** [10]). We will present variants of this observation: maximal bisimulation on different coalgebras i.e., not necessarily the final ones; and more generally arbitrary relations on elements of coalgebras.

The motivation for this work is the use of coinductive predicates in theorem proving. Already in the short history of coinductive theorem proving it has become clear that most interesting behavioural properties usually need relatively complex coinductive predicates. Evidence can be found in the attempts to verify protocols [7], modalities [5] or even basic metric predicates on streams [3, 4]. This indicates that bisimilarity alone is not powerful enough for proving many properties of infinite objects. There are tools for the automatic generation of bisimulation [12]. Usually in such tools all other behavioural properties (e.g. the examples in Section 3) will be translated into equational problems so that they can be tackled using bisimulations. Depending on the problem domain, these translations might be costly or cumbersome; our aim is to make such tools applicable to automatic proofs for a larger class of properties without having to reduce each such property to an equational problem. To be more precise we would like to generalise the well-known hidden-algebraic fact 'Behavioural equivalence is bisimilarity' [13, 6] to a larger class of relations.

We restrict ourselves to a class of endofunctors on **Set**. However this work can be read in two different ways, in a categorical model of dependent type theory or in **Set** and relations on them.

*Supported by VENI grant nr. 639.021.714 from the Netherlands Organisation for Scientific Research (NWO).

2 Relation Lifting

A very general coalgebraic method for defining bisimulation is using *relation lifting* [11, Ch. 3]. Let **Rel** be the category of binary relations and relation-preserving maps, i.e., maps that make the leftmost diagram below commute (here $f_1 \times f_2$ on top is the obvious restriction of the bottom one).

$$\begin{tikzcd}
R \arrow[r, "f_1\times f_2"] \arrow[d, hook] & S \arrow[d, hook] \\
X_1\times Y_1 \arrow[r, "f_1\times f_2"] & X_2\times Y_2
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& \mathbf{Rel}(F)(R) \arrow[dr, hook] & \\
F(R) \arrow[ur, two heads] \arrow[rr, "\langle F\pi_1, F\pi_2\rangle"] & & FX\times FY
\end{tikzcd}$$

Let $F\colon \mathbf{Set} \longrightarrow \mathbf{Set}$ be a functor. Then $\mathbf{Rel}(F)\colon \mathbf{Rel} \longrightarrow \mathbf{Rel}$, the *relation lifting of* F, is the functor taking a binary relation $\langle \pi_1, \pi_2\rangle\colon R \subseteq X \times Y$ to the image of $\langle F\pi_1, F\pi_2\rangle\colon FR \longrightarrow FX \times FY$ (see the rightmost diagram above). Given $f_1 \times f_2\colon R \longrightarrow S$ this **Rel**-functor is defined on morphisms as $\mathbf{Rel}(F)(f_1 \times f_2) := F(f_1) \times F(f_2)$. It is a well-known fact that relation lifting preserves equality, and thus it is geared towards proving equalities by constructing bisimulations. This make it unsuitable for working with arbitrary coinductive predicates. We consider a generalisation of relation lifting that can be used for a larger class of predicates.

Fix two sets X and Y. We denote by $\mathbf{Rel}_{XY}$ the lattice of subsets of $X \times Y$ considered as a subcategory of **Rel**, i.e., the objects are binary relations between X and Y and the morphisms are inclusion maps.

Let $\mathbf{rel}(F)\colon \mathbf{Rel}_{XY} \longrightarrow \mathbf{Rel}_{XY}$ be a *monotonic* functor such that for $R \subseteq X \times Y$ we have $\mathbf{rel}(F)(R) \subseteq F(X) \times F(Y)$. While the standard relation lifting takes R to a canonical subset $\mathbf{Rel}(F)(R) \subseteq F(X) \times F(Y)$, in our setting we deal with an arbitrary monotonic functor $\mathbf{rel}(F)$ taking relations on $X \times Y$ to relations on $F(X) \times F(Y)$. The reason is that the standard construction of bisimulations can, in a more generic way, be carried over to $\mathbf{rel}(F)$.

Note further that $\mathbf{Rel}(F)$ is an endofunctor on **Rel** while $\mathbf{rel}(F)$ is parametrised by X, Y and need not be defined globally. Obviously the theory we develop works for a $\mathbf{rel}(F)$ that is defined uniformly across **Rel**, so this is not a restriction. However, the local character of $\mathbf{rel}(F)$ allows the expression of finer properties. The examples in Section 3 demonstrate a globally defined $\mathbf{rel}(F)$ while in Section 4 we show examples where the local structure of $\mathbf{rel}(F)$ is needed.

For F-coalgebras $\alpha_X\colon X \longrightarrow FX$ and $\alpha_Y\colon Y \longrightarrow FY$, let $\tilde{F}_{XY}\colon \mathbf{Rel}_{XY} \longrightarrow \mathbf{Rel}_{XY}$ be the functor defined on objects as the inverse image of $\mathbf{rel}(F)$ alongside $\alpha_X \times \alpha_Y$ i.e.,

$$\tilde{F}_{\alpha_X\alpha_Y}(R) = \{\langle x, y\rangle \mid \langle \alpha_X(x), \alpha_Y(y)\rangle \in \mathbf{rel}(F)(R)\} \ ,$$

We usually drop the subscripts if they are understood from the context. As $\mathbf{rel}(F)$ is monotonic, so is $\tilde{F}$. In other words, $\tilde{F}$ is well-defined on the morphisms of $\mathbf{Rel}_{XY}$.

Note that in general we need not have $R \subseteq \tilde{F}(R)$ as this depends on the dynamics of α_X and α_Y. But if $R \subseteq \tilde{F}(R)$ then R with the inclusion map constitutes a $\tilde{F}$-coalgebra. Using Knaster-Tarski's fixed-point theorem, we can prove the following proposition.

Proposition 2.1. *The final coalgebra of* $\tilde{F}$ *exists in* $\mathbf{Rel}_{XY}$.

We denote the final $\tilde{F}$-coalgebras by $\nu\tilde{F}$. The final $\tilde{F}$-coalgebras correspond to coinductive predicates. This is because finality entails the equality

$$\tilde{F}(\nu\tilde{F}) = \nu\tilde{F} \ , \tag{1}$$

which means

$$\langle x, y\rangle \in \nu\tilde{F} \Leftrightarrow \langle \alpha_X(x), \alpha_Y(y)\rangle \in \mathbf{rel}(F)(\nu\tilde{F}) \ .$$

If one was to consider the above as the 'definition' of predicate $\nu\tilde{F}$ then the recursive occurrence of $\nu\tilde{F}$ as argument of $\mathbf{rel}(F)$ would supply the circularity evident in the coinductive predicates. Hence, by suitably choosing $\mathbf{rel}(F)$, α_X and α_Y, one can recover coinductive predicates as final $\tilde{F}$-coalgebras.

2.1 Bisimulation and Coinduction

Perhaps the most common example of a coinductive predicate is bisimulation between infinite objects. Any bisimulation between α_X and α_Y is a $\mathbf{Rel}(F)$-coalgebra $(R, \alpha_X \times \alpha_Y)$ in $\mathbf{Rel}_{\alpha_X \alpha_Y}$. Note that if we take $\mathbf{rel}(F) := \mathbf{Rel}(F)$ then $R \subseteq \tilde{F}(R)$ if and only if R is a bisimulation between α_X and α_Y [11]. This means that bisimulations are $\tilde{F}$-coalgebras with inclusion as transition map. Since $\tilde{F}$ is a monotonic functor it has a post fixed-point by Knaster-Tarski theorem, which is the maximal bisimulation [14]. The following proposition is then a reformulation of Proposition 2.1.

Proposition 2.2. *The final coalgebra of $\tilde{F}_{XY}$ exists in $\mathbf{Rel}_{\alpha_X \alpha_Y}$; its carrier is isomorphic with the maximal bisimulation between α_X and α_Y and its structure map is the identity inclusion map.*

Since equality includes any bisimulation relation on the carrier of an observable coalgebra (known also as a simple coalgebra) [14, Theorem 8.1] we have the following corollary.

Corollary 2.3 (Coinduction)**.**

i) Let $\langle \Omega^\circ, \alpha_{\Omega^\circ} \rangle$ be an observable F-coalgebra. If $\langle x, y \rangle \in \nu\tilde{F}_{\alpha_{\Omega^\circ} \alpha_{\Omega^\circ}}$ then $x = y$.

ii) Let $\langle \Omega, \alpha_\Omega \rangle = \nu F$ in **Set***. If $\langle x, y \rangle \in \nu\tilde{F}_{\alpha_\Omega \alpha_\Omega}$ then $x = y$.*

Part (ii) of the corollary above is the basis for *type theoretic coinduction* that is used is systems such as *Coq* [1]. There, instead of the usual bisimulation-building technique, one shows that $\langle x, y \rangle$ is an element of the final coalgebra by constructing $x = y$ as a canonical element of the final coalgebra. This is possible because of the isomorphism in (1) which allows one to construct canonical elements using sufficiently guarded specifications.

3 Pointwise Coinductive Predicates on Streams

In this section we present some examples on streams to demonstrate that generalising the definition of $\mathbf{Rel}(F)$ to $\mathbf{rel}(F)$ indeed enables us to define more predicates.

In [11, § 3.1] an alternative inductive definition is given for relation lifting of polynomial functors which coincides with the aforesaid definition for $\mathbf{Rel}(F)$. Based on that inductive definition constant functors are lifted to the equality on their range, i.e., $\mathbf{Rel}(\Lambda X.A)(R) = \Delta_A$. Compared with the work in [11] this is what we modify: we replace equality by $\star \subseteq A \times A$, an arbitrary binary relation on A.

Let $F(X) := \mathbf{2} \times X$ and $\nu F = (\mathbf{2}^\omega, \langle \mathtt{hd}, \mathtt{tl} \rangle)$ be the set of binary streams as a final coalgebra. Now assume a relation $\star \subseteq \mathbf{2} \times \mathbf{2}$ and define for any sets X, Y and $R \subseteq X \times Y$:

$$\mathbf{Rel}_\star(F)(R) = \{\langle \langle b_1, x \rangle, \langle b_2, y \rangle \rangle \mid \langle b_1, b_2 \rangle \in \star \wedge xRy\} \ .$$

Since $\mathbf{Rel}_\star$ is monotonic, by taking $\mathbf{rel}(F) := \mathbf{Rel}_\star(F)$ we can define the corresponding $\tilde{F}$ from Section 2 and apply Proposition 2.1 to get its final coalgebra. Clearly the description of this final coalgebra depends on the relation $\star$. For the case where the underlying coalgebras α_X, α_Y are the final coalgebra of streams

[1] In *Coq* and other intensional type theories this does not entail $x = y$ *inside* the system, but this issue is beyond the present paper.

this has a simple solution. The answer is given in the proposition below, for which we need some definitions. Define the binary relation $\circledast$ on streams as

$$\circledast := \{\langle \sigma, \tau \rangle \mid \forall n, \langle \mathtt{hd}(\mathtt{tl}^n(\sigma)), \mathtt{hd}(\mathtt{tl}^n(\tau)) \rangle \in \star\} \ ,$$

i.e., two streams σ, τ belong to $\circledast$ if and only if they are pointwise in relation $\star$. Note that

$$\begin{aligned} \tilde{F} \circledast &= \{\langle \sigma, \tau \rangle \mid \langle \langle \mathtt{hd}, \mathtt{tl} \rangle(\sigma), \langle \mathtt{hd}, \mathtt{tl} \rangle(\tau) \rangle \in \mathbf{Rel}_\star(F)(\circledast)\} \\ &= \{\langle \sigma, \tau \rangle \mid \langle \mathtt{hd}(\sigma), \mathtt{hd}(\tau) \rangle \in \star \wedge \langle \mathtt{tl}(\sigma), \mathtt{tl}(\tau) \rangle \in \circledast\} \end{aligned}$$

Note that $\circledast \subseteq \tilde{F} \circledast$; hence we can define $\alpha_\circledast : \circledast \longrightarrow \tilde{F} \circledast$ as the inclusion map. We have the following proposition.

Proposition 3.1. $\nu \tilde{F} = (\circledast, \alpha_\circledast)$.

The proof is a straightforward induction and resembles the proof of finality of the set of streams in [1].

If $R \subseteq \tilde{F}(R)$ we shall call R a $\star$-*simulation*. Clearly $\circledast$ is a $\star$-simulation on $\mathbf{2}^\omega$. Note that if R is $\star$-simulation and $\langle \sigma, \tau \rangle$ in R, then $\langle \mathtt{tl}^n(\sigma), \mathtt{tl}^n(\tau) \rangle \in R$ for all n. From this fact we can obtain the following principle.

Proposition 3.2 ($\star$-Coinduction)**.** *Relation $\circledast$ is the maximal $\star$-simulation relation on $\mathbf{2}^\omega$. I.e., in order to prove two streams are in $\circledast$ it suffices if we find a $\star$-simulation between them.*

Example 3.3.

i) Taking $\star := \Delta_\mathbf{2}$ we can recover relation lifting of [11], as well as bisimilarity and the coinduction proof principle.

ii) Taking $\star$ to be $\neq$ i.e., $\star := \{\langle 0,1 \rangle, \langle 1,0 \rangle\}$ we get the pointwise *inequality* between streams as a coinductive predicate. I.e., the $\circledast := \not\simeq$ where $\sigma \not\simeq \tau$ if and only if $\mathtt{hd}(\mathtt{tl}^n(\sigma)) \neq \mathtt{hd}(\mathtt{tl}^n(\sigma))$ for all n. Assume constant streams `zeros`, `ones` to be defined as

$$\begin{aligned} \mathtt{hd}(\mathtt{zeros}) &:= 0 \ , \quad \mathtt{tl}(\mathtt{zeros}) := \mathtt{zeros} \ ; \\ \mathtt{hd}(\mathtt{ones}) &:= 1 \ , \quad \mathtt{tl}(\mathtt{ones}) := \mathtt{ones} \ . \end{aligned}$$

Then since $R_{\neq} := \{\langle 0 :: \mathtt{zeros}, 1 :: \mathtt{ones} \rangle\}$ is a $\neq$-simulation by Proposition 3.2 we have $\mathtt{zeros} \not\simeq \mathtt{ones}$.

iii) Similar to above, by taking $\star := \{\langle 0,1 \rangle\}$ (resp. $\star := \{\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,1 \rangle\}$) we get $\prec$ the pointwise *less than* (resp. $\preccurlyeq$ *less or equal*) relation between streams as a coinductive predicate. Again one can show by Proposition 3.2

$$\mathtt{zeros} \prec \mathtt{ones} \ , \qquad \mathtt{zeros} \preccurlyeq \mathtt{ones} \ , \qquad \mathtt{zeros} \preccurlyeq \mathtt{zeros} \ .$$

Note that the relations in the example above are *not* the same as *simulations* in the sense of [10]. Simulation (and *lax* relation lifting), albeit itself a greatest fixed point [10, Lemma 5.1], is based on an order on the functor while in our pointwise comparison we use an arbitrary binary relation on data which is not necessarily an order relation.

Proposition 3.2 is useful in that it mimics the ordinary coinduction proof principle. However, one can also directly use the finality of $\circledast$ to construct its elements in their canonical form. This leads to another instance of type theoretic coinduction.

We conclude this section by pointing out that the above lifting of pointwise relations to the stream level can be done for relations with different arity. In fact it is straightforward to obtain the counterpart of Proposition 3.2 for n-ary relations. For example one can consider a ternary relation $\star_3 := \{\langle 0,0,0\rangle, \langle 0,1,1\rangle, \langle 1,0,1\rangle, \langle 1,1,1\rangle\}$ that leads to the relation $\curlyvee$ corresponding to the pointwise disjunction of streams. This is specially useful in proving properties in stream calculus [15], because there we deal with causal functions and examining the components pointwise will usually suffice.

4 Mixed Coinductive Predicates on Streams

The pointwise coinductive predicates, though useful in many cases, have a rather restricted shape that limits their expressiveness. While our Proposition 2.1 is quite general, it is not always easy to find a simple description of the elements of the final coalgebra as in Proposition 3.1. In this section, still working with binary streams, we show some more intricate coinductive predicates that are useful in practise. In particular we show two examples from [2] which are used in a coinductive timed stream semantics of channel-based coordination[2].

Fix nonempty relations $\star_1, \star_2 \subseteq \mathbf{2}\times\mathbf{2}$. Assume (X,α_X) and (Y,α_Y) are F-coalgebras. Let

$$\begin{aligned}\mathbf{Rel}_{\star_1\star_2}(F)(R) = &\{\langle\langle b_1,x\rangle,\langle b_2,y\rangle\rangle \mid \langle b_1,b_2\rangle \in \star_1 \wedge \exists t \in \alpha_Y^{-1}\langle b_2,y\rangle,\ \langle x,t\rangle \in R\} \ \cup \\ &\{\langle\langle b_1,x\rangle,\langle b_2,y\rangle\rangle \mid \langle b_1,b_2\rangle \in \star_2 \wedge \exists t \in \alpha_X^{-1}\langle b_1,x\rangle,\ \langle t,y\rangle \in R\} \ .\end{aligned}$$

Then $\mathbf{Rel}_{\star_1\star_2}(F)$ is a monotonic endofunctor on $\mathbf{Rel}_{XY}$. Again the corresponding $\tilde{F}$ and its final coalgebra can be formed according to Section 2, but this general form is too complex to be useful. The special case where $\alpha_X = \alpha_Y = \langle \mathtt{hd},\mathtt{tl}\rangle$ leads to some simplification. In that case,

$$\begin{aligned}\tilde{F}(R) = &\{\langle\sigma,\tau\rangle \ \mid\ \langle \mathtt{hd}(\sigma),\mathtt{hd}(\tau)\rangle \in \star_1 \wedge \langle \mathtt{tl}(\sigma),\tau\rangle \in R\} \ \cup \\ &\{\langle\sigma,\tau\rangle \ \mid\ \langle \mathtt{hd}(\sigma),\mathtt{hd}(\tau)\rangle \in \star_2 \wedge \langle \sigma,\mathtt{tl}(\tau)\rangle \in R\} \ .\end{aligned}$$

Instantiating with $\star_1 = \{\langle 0,1\rangle\}$, $\star_2 = \{\langle 1,0\rangle\}$ one can observe that the final coalgebra $\nu\tilde{F}$ consists of the set of binary streams that satisfy the $\bowtie$ relation as defined in [2]. In short, if $\sigma \bowtie \tau$ and if both σ and τ are interpreted as time streams corresponding to events on the two ports of a channel, then σ and τ are completely asynchronous. Again we call relation R a $\bowtie$-simulation if $R \subseteq \tilde{F}(R)$ and we will have a counterpart of Proposition 3.2. I.e., in order to prove that two time streams are asynchronous it suffices to find a $\bowtie$-simulation between them.

Next example concerns the *merge* connective in [2], which captures the behaviour of a merger channel with two inputs and one output, and merges its two input streams to form the output. Here we work with ternary relations.

$$\begin{aligned}\mathbf{Rel}_{merge}(F)(R) = &\{\langle\langle b_1,x\rangle,\langle b_2,y\rangle,\langle b_3,z\rangle\rangle \mid \langle b_1,b_2\rangle \in \star_1 \wedge \langle b_1,b_3\rangle \in \star_2 \wedge \exists t{\in}\alpha_Y^{-1}\langle b_2,y\rangle,\ \langle x,t,z\rangle \in R\} \ \cup \\ &\{\langle\langle b_1,x\rangle,\langle b_2,y\rangle,\langle b_3,z\rangle\rangle \mid \langle b_1,b_2\rangle \in \star_3 \wedge \langle b_2,b_3\rangle \in \star_4 \wedge \exists t{\in}\alpha_X^{-1}\langle b_1,x\rangle,\ \langle t,y,z\rangle \in R\} \ .\end{aligned}$$

Assuming $\alpha_{\_}$'s are all the structure map of the final coalgebra of streams we have

$$\begin{aligned}\tilde{F}(R) = &\{\langle\sigma_1,\sigma_2,\tau\rangle \mid \langle \mathtt{hd}(\sigma_1),\mathtt{hd}(\sigma_2)\rangle \in \star_1 \wedge \langle \mathtt{hd}(\sigma_1),\mathtt{hd}(\tau)\rangle \in \star_2 \wedge \langle \mathtt{tl}(\sigma_1),\sigma_2,\mathtt{tl}(\tau)\rangle \in R\} \ \cup \\ &\{\langle\sigma_1,\sigma_2,\tau\rangle \mid \langle \mathtt{hd}(\sigma_1),\mathtt{hd}(\sigma_2)\rangle \in \star_3 \wedge \langle \mathtt{hd}(\sigma_2),\mathtt{hd}(\tau)\rangle \in \star_4 \wedge \langle \sigma_1,\mathtt{tl}(\sigma_2),\mathtt{tl}(\tau)\rangle \in R\} \ .\end{aligned}$$

[2] Timed data streams in [2] have a data component as well. For brevity, here we do not tackle the data and only deal with the time. Another simplification is that we use binary time but one could repeat this for $F(X) = \mathbb{R}^+ \times X$.

Instantiating with $\star_1 = \{\langle 0,1\rangle\}$, $\star_3 = \{\langle 1,0\rangle\}$, $\star_2 = \star_4 = \Delta_{\mathbf{2}}$ we obtain a coinductive predicate describing the behaviour of the merger channel. Furthermore we obtain a notion of *merge*-simulation and a corresponding coinduction principle. This enables us to prove, by finding a *merge*-simulation, that three time streams correspond to events on the ports of a merger.

Our last example illustrates a predicate $\asymp$ on binary streams such that (denoting $\mathtt{hd}(\mathtt{tl}^n(\sigma))$ by σ_n):

$$\sigma \asymp \tau := \forall n, \sigma_{2n} \leq \tau_{2n+1} \leq \sigma_{2n+2} \ \wedge\ \tau_{2n} \leq \sigma_{2n+1} \leq \tau_{2n+2} \ .$$

Hence $\sigma \asymp \tau$ means that $\sigma_0 :: \tau_1 :: \sigma_2 :: \tau_3 \cdots$ and $\tau_0 :: \sigma_1 :: \tau_2 :: \sigma_3 \cdots$ are both non-decreasing streams. Compared to the previous examples the predicate $\asymp$ examines larger initial segments of the streams and hence our candidate for $\mathbf{rel}(F)$ should observe deeper iterations of coalgebra. Let

$$\begin{aligned}\mathbf{Rel}_{\asymp}(F)(R) = \{\langle\langle b_1,x\rangle,\langle b_2,y\rangle\rangle \mid\ & \alpha_X(x) = \langle b_1',x'\rangle\ \wedge \alpha_X(x') = \langle b_1'',x''\rangle\ \wedge \\ & \alpha_Y(y) = \langle b_2',y'\rangle\ \wedge \alpha_Y(y') = \langle b_2'',y''\rangle \implies \\ & b_1 \leq b_2' \leq b_1''\ \wedge\ b_2 \leq b_1' \leq b_2''\ \wedge\ \langle x'',y''\rangle \in R\}\ .\end{aligned}$$

Then $\mathbf{Rel}_{\asymp}(F)$ is monotonic and we can form $\tilde{F}$ and its final coalgebra. Assuming $\alpha_X = \alpha_Y = \langle \mathtt{hd},\mathtt{tl}\rangle$ we obtain

$$\tilde{F}(R) = \{\langle\sigma,\tau\rangle \mid \sigma_0 \leq \tau_1 \leq \sigma_2 \wedge \tau_0 \leq \sigma_1 \leq \tau_2 \wedge \langle \mathtt{tl}^3(\sigma), \mathtt{tl}^3(\tau)\rangle \in R\}\ .$$

Then $\langle\sigma,\tau\rangle \in \nu\tilde{F} \Leftrightarrow \sigma \asymp \tau$ and in order to prove that $\sigma \asymp \tau$ we should find a relation R such that $R \subseteq \tilde{F}(R)$ and that $\langle\sigma,\tau\rangle \in R$. As an example let the streams $\mathtt{zos}$, $\mathtt{ozs}$ be defined as

$$\begin{aligned}&\mathtt{hd}(\mathtt{zos}) := 0\ ,\quad \mathtt{hd}(\mathtt{tl}(\mathtt{zos})) := 1\ ,\quad \mathtt{tl}^2(\mathtt{zos}) := \mathtt{zos}\ ;\\ &\mathtt{hd}(\mathtt{ozs}) := 1\ ,\quad \mathtt{hd}(\mathtt{tl}(\mathtt{ozs})) := 0\ ,\quad \mathtt{tl}^2(\mathtt{ozs}) := \mathtt{ozs}\ .\end{aligned}$$

Then taking $R_{\asymp} := \{\langle\mathtt{zos},\mathtt{ozs}\rangle, \langle\mathtt{ozs},\mathtt{zos}\rangle\}$, and considering that by ordinary coinduction

$$\mathtt{zos} = 0 :: \mathtt{ozs}\ ,\quad \mathtt{ozs} = 1 :: \mathtt{zos}\ ,$$

we obtain $\tilde{F}(R_{\asymp}) = R_{\asymp}$. Hence $\mathtt{zos} \asymp \mathtt{ozs}$.

5 Conclusion & Further Work

The fact that we can describe such predicates as final coalgebras in **Rel** has more usage. By having a final model in hand coinductive proofs will essentially turn into finding functions between various final coalgebras. Hence we can use several type of coinductive definition schemes (e.g. coiteration, corecursion and their generalisations) for more complicated proofs. For example using the examples developed in Sections 3–4 one can prove

$$\begin{aligned}&\forall\sigma\tau,\ \sigma \prec \tau \implies \sigma \not\asymp \tau\\ &\forall\sigma_1\sigma_2\tau,\ \mathit{merge}(\sigma_1,\sigma_2,\tau) \implies \sigma_1 \bowtie \sigma_2\ .\end{aligned}$$

The first implication is in fact a function between final coalgebras $\circledast_< \to \circledast_{\neq}$ that can be defined using the ordinary coiteration scheme.

In the future, we plan to work on automating the generation of various types of $\star$-simulation relations in the tools that are used for automatic generation of bisimulations [12]. This requires a thorough reformulation of hidden-algebraic machinery of behavioural equivalence in a more general way. Recall that two streams are behaviourally equivalent if and only if $\mathtt{hd}(\mathtt{tl}^n(\sigma)) = \mathtt{hd}(\mathtt{tl}^n(\sigma))$ for all n; and that this implies bisimilarity. Looking back at the definition of $\circledast$ we observe that it captures a notion of 'behaviourally being in relation $\star$' which then will imply $\star$-similarity. Our aim is to make this more precise by working in the categorical models of hidden-algebra [6].

References

[1] P. Aczel. Algebras and coalgebras. In R. C. Backhouse, R. L. Crole, and J. Gibbons, eds., *Revised Lectures from Int. Summer School and Wksh. on Algebraic and Coalgebraic Methods in the Mathematics of Program Construction (Oxford, Apr. 2000)*, v. 2297 of *Lect. Notes in Comput. Sci.*, pp. 79–88. Springer, 2002.

[2] F. Arbab and J. J. M. M. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, eds., *Revised Selected Papers from 16th Int. Wksh. on Algebraic Development Techniques, WADT 2002 (Frauenchiemsee, Sept. 2002)*, v. 2755 of *Lect. Notes in Comput. Sci.*, pp. 34–55. Springer, 2003.

[3] Y. Bertot. Filters on coinductive streams, an application to Eratosthenes' sieve. In P. Urzyczyn, ed., *Proc. of 7th Int. Conf. on Typed Lambda Calculi and Applications, TLCA 2005 (Nara, Apr. 2005)*, v. 3461 of *Lect. Notes in Comput. Sci.*, pp. 102–115. Springer, 2005.

[4] Y. Bertot. Affine functions and series with co-inductive real numbers. *Math. Struct. in Comput. Sci.*, 17(1):37–63, 2007.

[5] V. Capretta. Common knowledge as a coinductive modality. In E. Barendsen, V. Capretta, H. Geuvers, and M. Niqui, eds., *Reflections on Type Theory, λ-Calculus, and the Mind: Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday*, pp. 51–61. Radboud University Nijmegen, 2007.

[6] C. Cîrstea. Coalgebra semantics for hidden algebra: Parameterised objects an inheritance. In F. Parisi-Presicce, ed., *Selected Papers from 12th Int. Wksh. on Algebraic Development Techniques, WADT '97 (Tarquinia, June 1997)*, v. 1376 of *Lect. Notes in Comput. Sci.*, pp. 174–189. Springer, 1997.

[7] E. Giménez. *Un Calcul de Constructions Infinies et son Application a la Verification des Systemes Communicants*. PhD thesis, Ecole Normale Supérieure de Lyon, 1996.

[8] P. Hancock and A. Setzer. Interactive programs and weakly final coalgebras in dependent type theory. In L. Crosilla and P. Schuster, eds., *From Sets and Types to Topology and Analysis: Towards Practicable Foundations for Constructive Mathematics*, v. 48 of *Oxford Logic Guides*, pp. 115–134. Oxford Univ. Press, 2005.

[9] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Inform. and Comput.*, 145(2):107–152, 1998.

[10] J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comput. Sci.*, 327(1–2):71–108, 2004.

[11] B. Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observations*. Book draft, 2005. Available at `http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`,

[12] D. Lucanu and G. Roşu. CIRC: a circular coinductive prover. In T. Mossakowski, U. Montanari, and M. Haveraaen, eds., *Proc. of 2nd Int. Conf. on Algebra and Coalgebra in Computer Science, CALCO 2007 (Bergen, Aug. 2007)*, v. 4624 of *Lect. Notes in Comput. Sci.*, pp. 372–378. Springer, 2007.

[13] G. Malcolm. Behavioural equivalence, bisimulation, and minimal realisation. In M. Haveraaen, O. Owe, and O.-J. Dahl, eds., *Selected Papers from 11th Wksh. on Specification Abstract Data Types, Joint with 8th Compass Wksh., ADT/COMPASS 1995 (Oslo, Sept. 1995)*, v. 1130 of *Lect. Notes in Comput. Sci.*, pp. 359–378. Springer, 1996.

[14] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[15] J. J. M. M. Rutten. A coinductive calculus of streams. *Math. Struct. in Comput. Sci.*, 15(1):93–147, 2005.

Lower Bound for Evaluation of $\mu\nu$ Fixpoint

Paweł Parys[*]
Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw
Banacha 2, PL-02-097 Warszawa, Poland
parys@mimuw.edu.pl

Abstract

We consider a fixpoint expressions $\mu y.\nu x.f(x,y)$ over the lattice $\{0,1\}^n$, where $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ is any monotone function. We study only algorithms for calculating these expressions using f only as a black-box: they may only ask for the value of f for given arguments. We show that any such algorithm has to do at least about n^2 queries to the function f, namely $\Omega\left(\frac{n^2}{\log n}\right)$ queries.

1 Introduction

Fast evaluation of fixpoint expressions is a key problem in the fixpoint theory. We consider a special form of expressions:

$$\mu x_d.\nu x_{d-1}\dots\mu x_2.\nu x_1.f(x_1,\dots,x_d)$$

(when d is even, and starting from νx_d when d is odd). We call such expression $\mu\nu(d,f)$. Moreover we consider these expressions only over the lattice $L=\{0,1\}^n$ with the order defined by $a_1\dots a_n \le b_1\dots b_n$ when $a_i \le b_i$ for all i. The function f is an arbitrary monotone function $f\colon L^d \to L$. Calculating the value of $\mu\nu(d,f)$ is already a very general problem. The problem of finding winning positions in a parity game may be reduced to it in polynomial time (where n corresponds to the game graph size and d to the number of priorities). The problem of solving parity games is polynomial time equivalent to the non-emptiness problem of automata on infinite trees with the parity acceptance conditions [4], and to the model checking problem of the modal μ-calculus (modal fixpoint logic) [3, 6].

Although these are very important problems and many people were working on them, no one could show any polynomial time algorithm. Our goal is the opposite—to prove some lower bound. It may be very difficult to show any algorithmic lower bound, especially because it is known that the problems are in NP∩co-NP. In such situation the only possibility is to reformulate the problem slightly, so that it becomes combinatorial. To achieve that we use a black-box model (or an oracle model) introduced in [1]. Instead of arbitrary algorithms, which could analyze for example a formula defining f, we consider only algorithms, that can only ask for values of f for given arguments. Moreover we are not interested in their exact complexity, only in the number of queries to the function f. In other words we consider decision trees: each internal node of the tree is labeled by an argument, for which the function f should be checked, and each its child corresponds to a possible value of f for that argument. The tree has to determine the value of the fixpoint expression $\mu\nu(d,f)$: for each path from the root to a leaf there is at most one possible value of $\mu\nu(d,f)$ for all functions which are consistent with the answers on that path. We are interested in the height of such trees, which justifies the following definition.

Definition 1. For any natural number d and finite lattice L we define $num(d,L)$ as the minimal number of queries, which has to be asked by any algorithm correctly calculating expression $\mu\nu(d,f)$ basing only on queries to the function $f\colon L^d \to L$.

[*] Partly supported by Polish government grant N206 008 32/0810.

The most basic method of evaluating fixpoint expression is to use the observation that $\mu x.g(x) = g^n(\perp)$; so it is enough to evaluate n times g on the previous result, starting from the minimal element $\perp$. To get ν instead of μ, one should start from $\top$ instead of $\perp$. This generalizes to d nested fixpoints $\mu\nu(d,f)$ and requires $O(n^d)$ queries to f; see [5]. For some time no better algorithm was known. Then an algorithm using only $O(n^{\lfloor d/2\rfloor+1})$ queries to f was shown in [8] and [1], which was rather a surprise. Recently some better algorithms for the modal μ-calculus and parity games were discovered, like [9] working in time $O(n^{d/3})$ or [7] working in time $n^{O(\sqrt{n})}$. However these two algorithms use parity games framework and do not translate to the black-box model. Here we see one of the limitations of our model: there may exist fast algorithm, which uses a definition of f in some tricky manner, but is unable to work when it can only evaluate f. The other limitation is that the number of monotone functions definable by a short formula is only single exponential, while the number of all monotone functions $f\colon \{0,1\}^{nd} \to \{0,1\}^n$ is double exponential. When we restrict only to functions definable by a short formula it is possible that less queries would be needed.[1] Beside of that, the following are very important questions (following [1]): how good may we do using f only as a black-box? Is the complexity of about $n^{d/2}$ queries optimal? What is the optimal number of queries? If the answer will be rather high, we will know that any fast algorithm for parity games and modal μ-calculus has to use different techniques. If the answer will be rather low, it may also give some fast algorithm. We write „may" because there is no implication in a formal sense: the decision tree with small number of queries may be very irregular and it may take a lot of time to compute what the next query should be.

In this paper we consider only the case $d = 2$. We show that $\Omega\left(\frac{n^2}{\log n}\right)$ queries are necessary in that case (which is almost n^2). Our result is the following.

Theorem 2. *For any natural n it holds* $num(2,\{0,1\}^n) = \Omega\left(\frac{n^2}{\log n}\right)$.

This result is a first step towards solving the general question, for any d. It shows that in the black-box model something may be proved. Earlier it was unknown even if for any d there are needed more than nd queries. Note that $num(1,\{0,1\}^n)$ is n and that in the case when all d fixpoint operators are μ (instead of alternating μ and ν) it is enough to do n queries. So the result gives an example of a situation where the alternation of fixpoint quantifiers μ and ν is provably more difficult than just one type of quantifiers μ or ν. Although it is widely believed that the alternation should be a source of algorithmic complexity, the author is not aware of any other result showing this phenomenon, except the result in [2].

The paper is organized as follows. In Section 2 we reduce the problem from the lattice $\{0,1\}^n$ to some more convenient lattice. In Section 3 we define a family of difficult functions f. In Section 4 we finish the proof of Theorem 2.

Acknowledgment. The author would like to thank Igor Walukiewicz for suggesting this topic and many useful comments.

2 Changing the Lattice

Instead of the lattice $\{0,1\}^n$ it is convenient to use a better one. Take the alphabet Γ_n consisting of letters a_i for $1 \le i \le \frac{n(n+1)}{2} + 1$ and the alphabet $\Sigma_n = \{0,1\} \cup \Gamma_n$. We introduce the following partial order on it: the letters a_i are incomparable; the letter 0 is smaller than all other letters; the letter 1 is bigger than all other letters. We will be considering sequences of n such letters, i.e. the lattice is Σ_n^n. The order on the sequences is defined as previously: $a_1 \dots a_n \le b_1 \dots b_n$ when $a_i \le b_i$ for all i.

[1] This is not the case for $d = 2$; functions used in our lower bound proof are all definable by a boolean formula of size polynomial in n.

We formulate a general lemma, which allows to change a lattice in our problem. For any two lattices L_1, L_2 we say that $h\colon L_1 \to L_2$ is a homomorphism, when it preserves the order, i.e. $x \leq y$ implies $h(x) \leq h(y)$.

Lemma 3. *Let L_1, L_2 be two finite lattices and $enc\colon L_1 \to L_2$ and $dec\colon L_2 \to L_1$ two homomorphisms such that $dec \circ enc = id_{L_1}$. Then $num(d, L_1) \leq num(d, L_2)$.*

Proof
In other words we should be able to use any algorithm calculating $\mu\nu(d, f)$ in L_2 to calculate $\mu\nu(d, f)$ in L_1. Let $f_1\colon L_1^d \to L_1$ be the unknown function in L_1. We define $f_2\colon L_2^d \to L_2$ as $f_2(x_1, \ldots, x_d) = enc(f_1(dec(x_1), \ldots, dec(x_d)))$. Note that f_2 is a monotone function if f_1 was monotone, since *enc* and *dec* preserve the order.

Let $\bot_1, \bot_2, \top_1, \top_2$ be the minimal and maximal elements in L_1 and L_2. For any $x \in L_1$ we have $\bot_2 \leq enc(x)$, so $dec(\bot_2) \leq dec(enc(x)) = x$, which means that $dec(\bot_2) = \bot_1$. Similarly $dec(\top_2) = \top_1$.

See that $dec(\mu\nu(d, f_2)) = \mu\nu(d, f_1)$. This is true, because these fixpoint expressions may be replaced by a term containing applications of f and minimal and maximal elements. This is done in a classic way, we replace the fixpoint operators by a iterated nesting. The minimal required number of iterations depends on the structure. Here we have only two structures, L_1 and L_2, so we may take the bigger of the two minimal numbers. Hence we may use the same term in L_1 and L_2, the difference is if we use f_1 or f_2, $\bot_1$ or $\bot_2$, $\top_1$ or $\top_2$. Then easy induction on the term structure shows that $dec(\mu\nu(d, f_2)) = \mu\nu(d, f_1)$, because $dec(\bot_2) = \bot_1$, $dec(\top_2) = \top_1$, $dec(f_2(x_1, \ldots, x_d)) = f_1(dec(x_1), \ldots, dec(x_d))$. So to find $\mu\nu(d, f_1)$ it is enough to find $\mu\nu(d, f_2)$, which may be found for any f_2 in $num(d, L_2)$ queries to f_2. To evaluate f_2 in our case it is enough to do one query to f_1. Hence $\mu\nu(d, f_1)$ may be found in $num(d, L_2)$ queries (or maybe less queries in some other way). □

For the lattice Σ_n^n we have the following result, from which Theorem 2 follows:

Lemma 4. *For any natural n it holds $num(2, \Sigma_n^n) \geq \frac{n(n+1)}{2}$.*

Proof (Theorem 2)
We will show how Theorem 2 follows from this lemma. Take k such that $\binom{2k}{k} \geq \frac{n(n+1)}{2} + 1$. From the Stirling formula follows that $\binom{2k}{k}$ grows exponentially in k, so we may have $k = O(\log n)$. Take $m = \left\lfloor \frac{n}{2k} \right\rfloor$. From Lemma 4 for m we see that $num(2, \Sigma_m^m) \geq \frac{m(m+1)}{2} = \Omega\left(\frac{n^2}{\log n}\right)$.

Now it is enough to use Lemma 3 to see that $num(2, \{0,1\}^n) \geq num(2, \Sigma_m^m)$. We need to define functions $enc\colon \Sigma_m^m \to \{0,1\}^n$ and $dec\colon \{0,1\}^n \to \Sigma_m^m$. Each letter from Σ_m will be encoded in a sequence of $2k$ letters from $\{0,1\}$ in the following way: 0 is translated to the sequence of $2k$ zeroes, 1 to the sequence of $2k$ ones, any of the letters a_i is translated to some sequence of $2k$ bits, in which exactly k bits are equal to 1. Because $n \geq m$ we have $\binom{2k}{k} \geq \frac{m(m+1)}{2} + 1$, so there are enough different such sequences to encode all letters. We use this encoding to define $enc(x)$: an i-th letter of x is encoded in the i-th fragment of $2k$ bits and the final $n - 2km$ bits are set to zeroes. On the other hand to read an i-th letter of the value of $dec(y)$, we look at the i-th fragment of $2k$ bits: when it corresponds to one of the letters a_i, this a_i is the result; otherwise the result is 0 or 1 depending on whether there are less than k ones in the sequence or not. Note that *dec* is defined on all sequences, not only on results of *enc*. It is easy to see that $dec(enc(x)) = x$ for any $x \in \Sigma_m^m$ and that both functions are homomorphisms (mainly because encodings of different letters a_i are incomparable). □

3 Difficult Functions

In this section we define a family of functions used in a proof of Lemma 4. A function $f_{z,\sigma}\colon \Sigma_n^{2n} \to \Sigma_n^n$ is parametrized by a sequence $z \in \Gamma_n^n$ (which will be the result of $\mu y.\nu x.f_{z,\sigma}(x,y)$) and by a permutation $\sigma\colon \{1,\dots,n\} \to \{1,\dots,n\}$ (which is an order in which the letters of z are uncovered). Note that z is from Γ_n^n, not from Σ_n^n, so it can not contain 0 or 1, just the letters a_i. Whenever z and σ are clear from the context, we simply write f. In the following the i-th element of a sequence $x \in \Sigma_n^n$ is denoted by $x[i]$. A pair z,σ defines a sequence of values $y_0,\dots,y_n$:

$$y_k[i] = \begin{cases} z[i] & \text{for } \sigma^{-1}(i) \le k \\ 0 & \text{otherwise.} \end{cases}$$

In other words y_k is equal to z, but with some letters covered: they are 0 instead of the actual letter of z. In y_k there are k uncovered letters; the permutation σ defines the order, in which the letters are uncovered. Using this sequence of values we define the function. In some sense the values of the function are meaningful only for $y = y_k$, we define them first (assuming $y_{n+1} = y_n$):

$$f(x,y_k)[i] = \begin{cases} 0 & \text{if } \forall_{j>i} x[j] \le y_{k+1}[j] \text{ and } x[i] \not\ge y_{k+1}[i] & \text{(case 1)} \\ y_{k+1}[i] & \text{if } \forall_{j>i} x[j] \le y_{k+1}[j] \text{ and } x[i] \ge y_{k+1}[i] & \text{(case 2)} \\ x[i] & \text{if } \exists_{j>i} x[j] \not\le y_{k+1}[j] & \text{(case 3).} \end{cases}$$

For any other node y we look for the lowest possible k such that $y \le y_k$ and we put $f(x,y) = f(x,y_k)$. When such k does not exists ($y \not\le z$), we put $f(x,y)[i] = 1$.

Lemma 5. *The function f is monotone and $\mu y.\nu x.f(x,y) = z$.*

Proof
First see what happens when we increase x: take $x' \ge x$. We want to have $f(x',y)[i] \ge f(x,y)[i]$ for each i. Whenever for x and x' we are in the same case of the function definition, it is OK. Also when for x we have an earlier case than for x' it is OK (in particular when for x we have case 2, it holds $x'[i] \ge x[i] \ge y_{k+1}[i]$). On the other hand it is impossible, that for x' we get an earlier case than for x (it is easy to see looking at the conditions for choosing a case). Also when $y \not\le z$, for both x and x' we get the same result 1.

Now see what happens, when we increase y: take $y' \ge y$. When for y' there is $y' \not\le z$, we get a result 1, which is bigger than anything else. Otherwise the values y_k and $y_{k'}$ chosen for y and y' satisfy $y_{k'} \ge y_k$, so also $y_{k'+1} \ge y_{k+1}$. The argumentation that in such case $f(x,y_{k'})[i] \ge f(x,y_k)[i]$ is identical as for the change of x.

To calculate the fixpoint expression, first see that $\nu x.f(x,y_k) = y_{k+1}$. It follows immediately from the definition: $f(y_{k+1},y_k) = y_{k+1}$ and for any $x > y_{k+1}$ we get $f(x,y_k) \ne x$, because $f(x,y_k)$ differs from x on the last position i where $x[i] > y_{k+1}[i]$, we get there $y_{k+1}[i]$ instead of $x[i]$. The main fixpoint satisfies $\mu y.\nu x.f(x,y) = y_n = z$, because $y_{k+1} > y_k$ for all $k < n$ and $y_{n+1} = y_n$. □

4 The Proof

Now we will show that at least $\frac{n(n+1)}{2}$ queries are needed to calculate $\mu y.\nu x.f(x,y)$, even if we allow as f only functions from our family. The problem can be considered as a game between two players, we call them an algorithm and an oracle. In each round the algorithm player asks a query to the function, after what the oracle player chooses an answer (which is consistent with the previous answers). The algorithm player wins if after $\frac{n(n+1)}{2} - 1$ steps each function consistent with the answers has the same value of $\mu y.\nu x.f(x,y)$. Otherwise the oracle player wins. We have to show a winning strategy for the oracle player.

First see informally what may happen. Consider first a standard algorithm evaluating fixpoint expressions. It starts from $y = y_0 = 0\dots0$ and $x = 1\dots1$. Then it repeats $x := f(x,y)$ until x stops changing, in which case $x = \nu x.f(x,y)$. For our functions it means that in each step the last 1 in x is replaced by the corresponding letter of y_1. The loop ends after n steps with $x = y_1$. Then the algorithm does $y := x$, $x := 1\dots1$, and repeats the above until y stops changing. For any $y = y_k$ the situation is very similar: in each step the last 1 in x is replaced by the corresponding letter of y_{k+1} (we may say that this letter is uncovered).

In fact, by choosing appropriate x the algorithm may decide which letter of y_{k+1} he wants to uncover, but always at most one. For the algorithm only the letter on which y_{k+1} differs from y_k is important, as he already knows all letters of y_k. However the difference may be on any position on which y_k has 0 (it depends on σ). The oracle player may choose this position in the most malicious way: whenever the algorithm player uncovers some letter, the oracle decides that this is not the letter on which y_k and y_{k+1} differs. So the algorithm has to try all possibilities (all positions on which y_k has 0), which takes $\frac{n(n+1)}{2}$ steps. He may also ask for some other y. It can give him any profit only if he accidentally guesses some letters of z. However the oracle may always decide that the guess of the algorithm is incorrect (that the value of z is different).

Now come to a more formal proof. We show a strategy for the oracle player. During the game we (the oracle player) keep a variable cur ($0 \leq cur < n$), which is equal to 0 at the beginning and is increased during the game. Intuitively it means how many letters of z are already known to the algorithm player. By s we denote the number of queries already asked (it increases by 1 after each query) and by s_{lok} the number of queries asked for this value of cur (it increases by 1 after each query and is reset to 0 when cur changes).

At every moment we keep a set F of functions consistent with all the answers till now (there may be more consistent functions, but each function in our set has to be consistent). The set will be described by a set of permutations Π and by sets of allowed values $A_i \subseteq \Gamma_n$, one for each coordinate $1 \leq i \leq n$. The sets should satisfy the following conditions:

1. for each $i \leq cur$ there is only one value of $\sigma(i)$ for $\sigma \in \Pi$;
2. in Π there are permutations σ with at least $n - cur - s_{lok}$ different values of $\sigma(cur+1)$;
3. for each permutation $\sigma \in \Pi$ when we take any other permutation σ' which agrees with σ on the first $cur+1$ arguments ($\sigma(i) = \sigma'(i)$ for each $1 \leq i \leq cur+1$), we have $\sigma' \in \Pi$ as well;
4. for each $\sigma \in \Pi$ and $i \leq cur$ there is only one value in $A_{\sigma(i)}$ (note that thanks to condition 1, the value $\sigma(i)$ does not depend on the choice of σ);
5. for each $\sigma \in \Pi$ and $i > cur$ there are at least $\frac{n(n+1)}{2} + 1 - s$ values in the set $A_{\sigma(i)}$ (note that the set $\{\sigma(i) : i > cur\}$ does not depend on the choice of σ, as $\sigma(i)$ for $i \leq cur$ are fixed).

In the set F there are all functions $f_{z,\sigma}$ for which $\sigma \in \Pi$ and $z[i] \in A_i$ for each i. We see that in particular at the beginning all functions are in the set F. Note, that at each moment the value of y_{cur} is fixed, i.e. is the same for all functions in F (because $\sigma(i)$ and $z[\sigma(i)]$ are fixed for $i \leq cur$).

Now we specify how the answers are done for a query x,y. Whenever $y \leq y_i$ for some $i < cur$, we answer according to all the functions in our set F. The answer of each function is the same, as it depends only on the value of y_{i+1} (for the smallest i such that $y \leq y_i$), which is already the same for all functions. Such question does not give any new knowledge to the algorithm player.

Whenever $y \not\leq y_{cur}$, we remove the value $y[i]$ from the set A_i (only if it was there, in particular only if $y[i] \in \Gamma_n$) for each i such that $\sigma^{-1}(i) > cur$ for any $\sigma \in \Pi$ (note that once again this condition is satisfied for exactly the same i for every permutation in Π). All the conditions of F are still satisfied, as we removed only one value from the sets A_i after one additional query was done. In other words we remove all functions $f_{z,\sigma}$, in which $z[\sigma(i)] = y[\sigma(i)]$ for some $i > cur$. Then for each function from F we have $y \not\leq z$ (if $y \leq z$ then $y[i] = 0$ for each i with $\sigma^{-1}(i) > cur$, which means that $y \leq y_{cur}$). So we reply to the query by a sequence of ones, which is the case for all the functions in F. Intuitively this case talks about a situation when someone tries to guess z (or its part) instead of gently asking for $y = y_{cur}$. We prefer to answer that his guess was incorrect and to eliminate all functions with z similar to the y about which he asked.

Consider now the case when $y \leq y_{cur}$ but $y \not\leq y_{cur-1}$. Let ask be the greatest number such that $x[ask] \not\leq y_{cur}[ask]$ (if there is no such number we take $ask = 0$). Intuitively the algorithm player asks whether $\sigma(cur+1) = ask$; we prefer to answer NO, so he will have to try all the possibilities until he will discover the value of $\sigma(cur+1)$. The first case is when in Π there are permutations with $\sigma(cur+1) \neq ask$. Note that this is true at least $n - cur - 1$ times for this cur due to condition 2. In such case we remove from Π all the permutations with $\sigma(cur+1) = ask$ and we answer according to all the functions left in F. We have to argue that for each of them the answer is the same. On positions $i < ask$ there is always case 3, because $x[ask] \not\leq y_{cur}[ask] = y_{cur+1}[ask]$ (the equality is true, because $\sigma(cur+1) \neq ask$). On the positions $i \geq ask$ there is $x[j] \leq y_{cur}[j] \leq y_{cur+1}[j]$ for $j > i$, so we fall into the first two cases. For $i = ask$ the result depends only on $y_{cur+1}[ask]$ which for all functions is equal to $y_{cur}[ask]$. Consider positions $i > ask$. When $x[i] = 0$ the answer is 0 in both cases 1 and 2 (we may get case 2 only when $y_{cur+1}[i] = 0$). When $x[i] > 0$ it has to be $y_{cur+1}[i] = x[i]$, as $x[i] \leq y_{cur}[i] \leq y_{cur+1}[i] \neq 1$, we get case 2 and we answer $y_{cur+1}[i] = x[i]$.

The last case is when all the permutations $\sigma \in \Pi$ have $\sigma(cur+1) = ask$. Then we choose any letter from A_{ask} and we remove all other letters from A_{ask}. In other words y_{cur+1} becomes fixed, so answers for all the functions left in F are the same. We increase cur (when cur becomes equal to n, we fail). It is easy to see, that all the conditions on the set F are still satisfied.

As already mentioned, before the last case holds there has to be $n - cur - 1$ earlier queries for this cur (we increase cur after at least $n - cur$ queries), so before $\frac{n(n+1)}{2}$ queries there is no danger that cur becomes equal to n. Moreover we are sure that in F there are functions with two different value of z (which is the result of the fixpoint expression): it is enough to take any σ from Π and then in $A_{\sigma(n)}$ there are at least two values ($z[\sigma(n)]$ may be equal to both of them).

References

[1] A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178(1–2):237–255, 1997.

[2] A. Dawar and S. Kreutzer. Generalising automaticity to modal properties of finite structures. *Theor. Comput. Sci.*, 379(1–2):266–285, 2007.

[3] E. A. Emerson. Model checking and the mu-calculus. In N. Immerman and P. G. Kolaitis, eds., *Proc. of DIMACS Wksh. on Descriptive Complexity and Finite Models (Princeton Univ., Jan. 1996)*, v. 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 185–214. AMS, 1997.

[4] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In C. Courcoubetis, ed., *Proc. of 5th Int. Conf. on Computer-Aided Verification, CAV '93 (Elounda, June/July 1993)*, v. 697 of *Lect. Notes in Comput. Sci.*, pp. 385–396. Springer, 1993.

[5] E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *Proc. of 1st Ann. IEEE Symp. on Logic in Computer Science, LICS '86 (Cambridge, MA, June 1986)*, pp. 267–278. IEEE CS Press, 1986.

[6] E. Grädel, W. Thomas, and T. Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research*, v. 2500 of *Lect. Notes in Comput. Sci.* Springer, 2002.

[7] M. Jurdzinski, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. on Comput.*, 38(4):1519–1532, 2008.

[8] D. E. Long, A. Browne, E. M. Clarke, S. Jha, and W. R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. In D. L. Dill, ed., *Proc. of 6th Int. Conf. on Computer-Aided Verification, CAV '94 (Stanford, CA, June 1994)*, v. 818 of *Lect. Notes in Comput. Sci.*, pp. 338–350. Springer, 1994.

[9] S. Schewe. Solving parity games in big steps. In V. Arvind and S. Prasad, eds., *Proc. of 27th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2007 (New Delhi, Dec. 2007)*, v. 4855 of *Lect. Notes in Comput. Sci.*, pp. 449–460. Springer, 2007.

A Bilattice Based Fixed Point Semantics for Integrating Imperfect Information

Daniel Stamate
Department of Computing, Goldsmiths, University of London
Lewisham Way, New Cross, London SE14 6NW, United Kingdom
d.stamate@doc.gold.ac.uk

Abstract

We present an approach to reasoning non-uniformly by default with uncertain, incomplete and inconsistent information using sets of rules/extended logic programs in the context of logics with a bilattice structure. A fixed point semantics for extended logic programs used in the process of inference is described, along with its computational approach. We show how this theoretic approach is applicable to the problem of integration of imperfect information coming from multiple sources.

1 Introduction

Information integration has received much attention for a number of year now in Database, Artificial Intelligence, Logic Programming, Multimedia Information Systems, World Wide Web and other research communities. Various approaches to information fusion have been proposed, adapted to the particular research areas, as the integration of data in a distributed database or from different databases, or the integration of information collected by an agent from other sources, or merging belief bases represented using logic programs, or integrating information coming from different medium sources as text, sound or image (as it is the case, for instance, in the query ”find the full description of albums containing music played by piano and having elements of classical and jazz as genre”) or Web sources, etc.

In order to propose an approach to information integration, two main questions may arise: (1) How is information coming from multiple sources combined?, and (2) Given the problems of possible conflicting information coming from mutually contradictory sources, of missing information coming from incomplete sources, or of uncertain information coming from sources of limited reliability, what meaning can one assign to the fused information (that is, what is the result of the integration)? The information that is incomplete or totally or partially inconsistent or uncertain will be called imperfect information in what follows.

With respect to the first question, the approach to the information integration that we propose in this paper is based on the logic programming paradigm, as it uses inference rules to integrate information in a logic based context. The logic rules we use, however, form extended logic programs as there is a need to employ, apart operations as the conjunction $\wedge$, the disjunction $\vee$ and the negation $\neg$, two more operations, that can be easily given a particular meaning in information merging, called the consensus $\otimes$ and the collecting together operation $\oplus$, to be formally and most generally defined in the next section.

With respect to the second question, we first choose an appropriate formalism based on multiple valued logics expressed by the concept of bilattice, that is very powerful in expressing the three aspects of imperfect information, namely the uncertainty, the incompleteness and the inconsistency.

In order to illustrate the concept of bilattice, assume first that we want to express the truthness of an information A. In the ideal case we can employ the logical values true or false, but in many situations this approach is simplistic and not acceptable. If we use a degree between 0 and 1 instead of a classical logical value, the approach is more appropriate in expressing uncertainty but less helpful in expressing lack of information, or the presence of contradiction in information. Indeed, no value from [0,1] can express, alone, incompleteness or inconsistency. A natural idea would then be to assign an information a pair $\langle c,d\rangle$ instead of one value, that would consist in a degree of confidence c and a degree of doubt d in [0,1], which do not necessarily add up to 1 (otherwise the single value c would suffice and we would be again in the previous case). In this setting $\langle 0,1\rangle$ and $\langle 1,0\rangle$, represent no confidence, full doubt, and full confidence, no doubt, so they would correspond to the classical values $false$ and $true$, respectively. On the other hand $\langle 0,0\rangle$ and $\langle 1,1\rangle$, represent no confidence, no doubt, and full confidence, full doubt, and they express a total lack of information or a total inconsistency, respectively. Two orders, namely the truth and the information (or knowledge) orders denoted $\leq_t$ and $\leq_i$, can naturally be defined on the set of confidence-doubt pairs, denoted $\mathscr{L}^{\mathscr{CD}}$ and called the confidence-doubt logic [9], as follows: $\langle x,y\rangle \leq_t \langle z,w\rangle$ iff $x\leq z$ and $w\leq y$, and $\langle x,y\rangle \leq_i \langle z,w\rangle$ iff $x\leq z$ and $y\leq w$, where $\leq$ is the usual order between reals. Intuitively speaking, an increase in the truth order corresponds to an increase in the degree of confidence and a decrease in the degree of doubt, while an increase in the information order corresponds to an increase in both degrees of confidence and doubt. The meet and join operations w.r.t. $\leq_t$ and $\leq_i$ are denoted $\wedge$, $\vee$, $\otimes$ and $\oplus$, respectively. $\wedge$ and $\vee$ are the extensions of the classical conjunction and disjunction, while $\otimes$ and $\oplus$ are two new operations with potential of use in information integration as they naturally express the idea of consensus and of collecting together of two pairs of confidence-doubt degrees. A natural extension of the classical negation is defined by $\neg\langle x,y\rangle = \langle y,x\rangle$. We conclude the section by noting that the double structure of lattice induced by the two orders on $\mathscr{L}^{\mathscr{CD}}$ is the basis of the general concept of bilattice introduced in [5].

2 Extended Logic Programs on Bilattices

Bilattices offer one of most capable frameworks to express, in the same time, the characteristics of the information to be incomplete, totally or partially inconsistent or uncertain. In addition, bilattices have an algebraic structure that allows to express approaches built on this concept in an elegant manner, and to facilitate elegant and often shorter proofs of results.

Definition 1. *A bilattice is a triple $\langle\mathscr{B},\leq_t,\leq_i\rangle$, where $\mathscr{B}$ is a nonempty set, and $\leq_t$ and $\leq_i$ are partial orders each giving $\mathscr{B}$ the structure of a complete lattice.*

Given the bilattice $\mathscr{B}$, join and meet operations under $\leq_t$ are denoted $\vee$ and $\wedge$, called extended disjunction and conjunction, and join and meet operations under $\leq_i$ are denoted $\oplus$ and $\otimes$, called collecting together and consensus, respectively. The greatest and least elements under $\leq_t$ are denoted $true$ and $false$, and the greatest and least elements under $\leq_i$ are denoted $\top$ and $\bot$. A bilattice has a negation, denoted $\neg$, if $\neg$ is a unary operation which is antimonotone w.r.t. the truth order and monotone w.r.t. the information order. In addition $\neg true = false$, $\neg false = true$, $\neg\bot = \bot$ and $\neg\top = \top$.

Note that $\mathscr{L}^{\mathscr{CD}}$ that we described in the previous section is a bilattice whose binary operations can be expressed as follows:

$$\langle x,y\rangle \wedge \langle z,w\rangle = \langle min(x,z), max(y,w)\rangle, \qquad \langle x,y\rangle \vee \langle z,w\rangle = \langle max(x,z), min(y,w)\rangle,$$
$$\langle x,y\rangle \otimes \langle z,w\rangle = \langle min(x,z), min(y,w)\rangle, \qquad \langle x,y\rangle \oplus \langle z,w\rangle = \langle max(x,z), max(y,w)\rangle.$$

In what follows we consider only bilattices for which all the distributive laws hold, an example of whom is $\mathscr{L}^{\mathscr{CD}}$. These bilattices are called distributive, and it was proven that their non-unary operations of finite or infinite arity, are monotone w.r.t. both the truth and the information orders [2].

Fitting [2] extended the notion of logic program, that we will call *extended program*, to bilattices as follows. Let $\mathscr{B}$ be a bilattice, whose elements will be referred to as logical values.

Definition 2. *(1) A formula is an expression built up from literals and elements of $\mathscr{B}$, using $\wedge,\vee,\otimes,\oplus,\neg,\exists,\forall$. (2) A rule r is of the form $H(v_1,...,v_n) \leftarrow F(v'_1,...,v'_m)$ where the atomic formula $H(v_1,...,v_n)$ is the head,* *and the formula $F(v'_1,...,v'_m)$ is the body. It is assumed that the free variables of the body are among $v_1,...,v_n$. (3) A program is a finite set of rules assuming that no predicate letter appearing in the head of more than one rule.*

Note that the restrictions/assumptions from (2) and (3) in the above definition cause no loss of generality, since any program as above, but without these restrictions, can be rewritten into another program respecting the restrictions [2]. On the other hand, any classical logic program can be written in the form described in definition 2, if one employs $\wedge$, $\vee$ and *true* only, from the operations and elements of the bilattice $\mathscr{B}$, which obviously embeds the classical bivalued logic.For technical reasons, from now on, we consider any extended program to be instantiated (that is, all the free variables are replaced by ground terms). Note that, due to the way extended programs have been defined, their instantiated versions have no more than one rule with the same head.

Example 1. *Consider the following set of rules / extended program in the context of the bilattice $\mathscr{L}^{\mathscr{C}\mathscr{D}}$.*

$A \leftarrow B \oplus F$; $B \leftarrow \neg E$;
$D \leftarrow B \vee C$; $E \leftarrow \langle 0.7, 0.3 \rangle$;
$C \leftarrow C \otimes E$.

Intuitively speaking, the information represented by E is assigned a confidence of 0.7 and a doubt of 0.3, as this fact is specified. B is the contrary of this information so it is assigned a confidence of 0.3 and a doubt of 0.7. F is an information whose confidence and support cannot be derived from the program as there is no rule defining F, so we assign F a confidence and doubt given by the reliability of the source providing it. That is, a default confidence and support will be assigned to F, specific to that source, as for instance a confidence of 0 and a doubt of 1 in a pessimistic, "not believing that source at all", approach, or a confidence of 1 and a doubt of 0 in an optimistic, "believing that source", approach. Let us assume the pessimistic approach for this source. C is information that should agree with E, as their consensus should be C. In particular C can be assigned a degree 0 of confidence and a degree 0 of doubt, if we are completely skeptical about the reliability of its source (that is, that source is not considered reliable, nor unreliable, so any default degrees of confidence and doubt from an information coming from that source will be 0 and 0 respectively). D is the information that is assigned the largest confidence and the least doubt from the confidence and doubt degrees of B and C, so 0.3 and 0, respectively. Note that C and D are incomplete as their degrees of confidence and doubt add up to less than 1. Finally A collects together the information B and F, so it will be assigned a confidence of 0.3 and a doubt of 1, so A is a partially inconsistent information as its confidence and doubt degrees add up to more than 1.

Roughly speaking, the example above illustrates the computation of the meaning/semantics of an extended program. In particular, the atoms (representing information to be integrated by rules) are assigned logical values from the underlying bilattice, process that needs the concept of interpretation and default interpretation.

Formally speaking, an interpretation is a mapping that assigns a logical value to each atom from the Herbrand base. In particular, if an atom cannot be derived from the rules then a default value, not necessarily the same for all atoms, is assigned to it. This default value is related to the degrees of reliability of the sources the information represented by the atom comes from. For instance if a source has a reliability of 90 percent, then the atom A, representing information coming from the source, and not being derived by any rule, is assigned by default a confidence of 0.9 and a doubt of 0.1. Formally speaking, a default interpretation is an interpretation. It is to be used to compensate the incompleteness of information derived using the program rules.

The derivation of information intuitively illustrated in the example above, will be formalised in the following section. In particular, as it is the case in most logic programming based approaches, the information deduction process will be expressed in terms of application of operators specific to the program, until no new information is obtained, that is, until a fixed point is reached. However, in our framework one should take into account also the particularities of the underlying logic provided by a set of values ordered w.r.t. a truth order and an information order, and the process of deduction by default regarding an atom when there is no rule to apply for that atom.

3 Program Operators and Fixed Point Semantics

The following defines the order $\leq_p$ and naturally extends the truth and information orders to the set of interpretations denoted by Int_P.

Definition 3. *If I and J are interpretations then*
(1) $I \leq_t J$ if $I(A) \leq_t J(A)$
(2) $I \leq_i J$ if $I(A) \leq_i J(A)$
(3) $I \leq_p J$ if $I(A) \neq \perp$ implies $I(A) = J(A)$
for any ground atom A.

The interpretations can be extended to closed formulas (i.e. formulas not containing free variables) as follows: $I(X \wedge Y) = I(X) \wedge I(Y)$, and similarly for the other operations of $\mathscr{L}^{\mathscr{C}\mathscr{D}}$, $I((\exists x)F(x)) = \bigvee_{s \in GT} I(F(s))$, and $I((\forall x)F(x)) = \bigwedge_{s \in GT} I(F(s))$, where GT stands for the set of all ground terms. If $I(B) = \beta$ we say that the formula B evaluates to the logical value β with respect to I. However, in some cases we can find out the value a closed formula evaluates to, no matter if some atoms are assigned the value $\perp$ - let us call them underdefined, thus the following concept:

Definition 4. *The closed formula B ultimately evaluates to the logical value β w.r.t. interpretation I, denoted by $B \equiv_I \beta$, if $J(B) = \beta$ for any interpretation J s.t. $I \leq_p J$.*

Let $I_\top$ be the interpretation obtained from I by assigning the value $\top$ to any underdefined atom. We have $B \equiv_I \beta$ iff $I(B) = I_\top(B) = \beta$.

The first inference operator assigned to an extended program P, called the *production operator* denoted Φ_P and defined below, intuitively corresponds to the activation of the program rules:

$$\Phi_P(I)(A) = \beta \text{ if } (\exists A \leftarrow B \in P \text{ and } B \equiv_I \beta), \text{ or}$$
$$\perp, \text{ otherwise.}$$

The second type of inference assumes the use of a fixed interpretation $\mathscr{D}$ called the *default interpretation*. Roughly speaking, the value of each atom A in the default interpretation is seen as being derived from the reliability degrees of the sources the information represented by A is coming from. For instance if two sources consisting in two medical studies found evidence, based on statistical tests with a 0.05 significance level, that medication m is effective in treating ailment a, while the other that the evolution of the ailment a is independent of whether or not the medication m was administered to the tested patients, then the atom $Effective(m,a)$ would be assigned the partially inconsistent value $\langle 0.95, 0.95 \rangle$ in the interpretation $\mathscr{D}$. This value will be used whenever no other value can be inferred for this atom from the program.

We introduce now an intermediary operator called the *refining operator*, denoted by Ψ_P, whose role is to refine an arbitrary default information X (part of the interpretation $\mathscr{D}$), in the sense that X either has to safely complete the information I obtained by activating the rules, in which case X is not modified by Ψ_P, or has to be modified into a new interpretation $\Psi_P(X,I)$ that safely completes I. Formally,

$$\Psi_P(X,I) = Rev(X, \Phi_P(Rev(X,I) \oplus I))$$

where $Rev(X,J)$ is an interpretation X' s.t. $X'(A) = X(A)$ for any ground atom A for which either $J(A) = \perp$ or $X(A) = J(A)$, and $X'(A) = \perp$ for any other ground atom A. We say that X' is the revision of X w.r.t. J.

Note that, by employing the refining operator, we wish to obtain the "best" default information X used to complete the interpretation I. Formally, we have the following requirements: (1) $X \leq_p \mathscr{D}$; (2) $X = \Psi_P(X,I)$; and (3) under the previous two conditions X is maximal w.r.t. $\leq_p$. Roughly speaking X is to be a part of the default interpretation $\mathscr{D}$ (condition 1), that, when it is revised by the sure information encoded in I and then is further revised by the information that is deducted using the rules applied to I completed with X, it is stable, that is, it does not change to refinement (condition 2). In addition X is supposed to complete as much as possible the information encoded in I (condition 3). That is, we are interested in the maximal fixed points of the operator $\lambda X \Psi_P(X,I)$ that are parts of $\mathscr{D}$, which we call *actual default interpretations* with respect to I and $\mathscr{D}$. We show below, via algebraic methods, that there exists a unique actual default interpretation w.r.t. I and $\mathscr{D}$.

Let $(S,\leq)$ be a complete semilattice. We define a diagonal contraction on S as being a binary operator $T' : S^2 \to S$ that satisfies $T'(X,X) \leq X$ for any $X \in S$. We provide the following useful lemmas.

Lemma 1. *If T is a monotone operator defined on the complete semilattice $(S,\leq)$ the following hold:*
(1) T has a least fixed point w.r.t. $\leq$.
(2) if Y is an element of S s.t. $T(Y) \leq Y$ then T has a greatest fixed point X below Y. Moreover X can be obtained as the limit of the following sequence: $X_0 = Y$, $X_n = T(X_{n-1})$ if n is a successor ordinal and $T(X_n) = inf_{\leq, m<n} T(X_m)$ if n is a limit ordinal.

Lemma 2. *Let T' be a binary operator defined on the complete semilattice $(S,\leq)$ which is monotone in its first argument and antimonotone in its second argument and is a diagonal contraction. If Y is an arbitrary element of S then T' has a greatest fixed point $X = T'(X,X)$ below Y. Moreover X can be obtained as the limit of the following sequence: $X_0 = Y$, $X_n = T'(X_{n-1}, X_{n-1})$ if n is a successor ordinal and $T(X_n) = inf_{\leq, m<n} T'(X_m, X_m)$ if n is a limit ordinal.*

We have the following properties of the production and the refining operators.

Proposition 1. *Φ_P is monotone w.r.t. $\leq_i$ and $\leq_p$ orders.*

Let $\Theta(X,Y,I) = Rev(X, \Phi_P(Rev(Y,I) \oplus I))$. Obviously $\Psi_P(X,I) = \Theta(X,X,I)$. We have the following:

Lemma 3. *$(\lambda X, \lambda Y)\Theta(X,Y,I)$ is monotone in its first argument and antimonotone in its second argument and is a diagonal contraction w.r.t. $\leq_p$.*

As a consequence of Lemmas 2 and 3 we get:

Proposition 2. *$(\lambda X)\Psi_P(X,I)$ has a greatest fixed point below $\mathscr{D}$ w.r.t. $\leq_p$, denoted by $Def_P^{\mathscr{D}}(I)$.*

Note that Proposition 2 involves that $Def_P^{\mathscr{D}}(I)$ is the unique actual default interpretation, while Lemma 2 provides a means of computation for $Def_P^{\mathscr{D}}(I)$ consisting in starting with the default interpretation $\mathscr{D}$ and iterating the operator $(\lambda X)\Psi_P(X,I)$ until a fixed point is reached. We call $Def_P^{\mathscr{D}}$ the *default operator*, as it is obvious that it reflects the application of the inference by default.

The two types of inference described above are now combined via a new operator, denoted Γ_P and called the *integrating operator*. Formally $\Gamma_P(I) = \Phi_P(I) \oplus Def_P^{\mathscr{D}}(I)$. Roughly speaking, given an interpretation I encoding the information inferred from the program so far, $\Gamma_P(I)$ encodes the new information currently inferred from the program.

Roughly speaking, in order to generate the information that can be derived from the extended program P we start with the least degree of information characterized by an interpretation I_0 in which all ground atoms are underdefined, denoted by $Const_\perp$ (i.e. nothing is known). We apply the two types of inference to the current information, which corresponds to an application of Γ_P operator, and we get a new interpretation I_1. This process is continued until nothing changes, that is, until a fixed point is reached. Formally we define the sequence $\mathscr{S}$ as follows:

$I_0 = Const_\perp$,
$I_n = \Gamma_P(I_{n-1})$ for a successor ordinal $n \geq 1$,
$I_n = inf_{\leq_p, m<n} I_m$ for n a limit ordinal.

We have:

Theorem 1. *The following hold:*
(1) $\mathscr{S}$ is increasing w.r.t. $\leq_p$ order (and thus w.r.t. $\leq_i$) and reaches a limit denoted by s.
(2) $\Gamma_P(s) = s$
(3) for any x s.t. $\Gamma_P(x) = x$ we have $s \leq_i x$.

Thus s is the least fixed point of Γ_P, and represents the minimal information that can be inferred from the extended program P completed with the default information $\mathscr{D}$. We chose s to designate the semantics of P. Note that any fixed point of Γ_P is deductively closed w.r.t. the program P and the default interpretation $\mathscr{D}$, and the two types of inference. Computationally speaking, we have:

Proposition 3. *If the program P does not contain any functional symbol, the semantics of P can be generated by iterative application of the integration operator in a finite number of steps , even if the underlying bilattice is infinite.*

Lemma 4. *If $Values(P)$ is the set of logical values appearing in the program P, and Closure(S) is the closure of the set of logical values from a subset S of the bilattice $\mathscr{B}$, to which one adds the elements $true$, $false$, $\top$, and $\perp$, w.r.t. the negation and the finite and infinite join and meet operations of $\mathscr{B}$, then $Closure(Values(P))$ is a finite bilattice.*

The proof of Proposition 3 is based on Theorem 1 and Lemma 4. Indeed, note that the logical values of any atom in the process of the computation of the semantics of P are elements of $Closure(Values(P))$, and are obtained as an increasing sequence w.r.t. $\leq_p$, and thus the evaluation of the semantics of P finishes in a finite number of steps since the Herbrand base is also finite.

4 Related Work

Our approach can be related in the first instance to other works regarding reasoning under uncertainty based on multivalued logics, in particular on bilattices, as those authored by Fitting. [3] defined the (multivalued) stable models for extended programs in bilattices that generalize the concept of stable models in the conventional bivalued logic [4]. We show below that if we consider the default interpretation $\mathscr{D}$ assigning the value $false$ to any ground atom, the semantics of P defined by our approach coincides with Fitting's multivalued stable model that has the least degree of information:

Proposition 4. *Let P be an extended program considered on the bilattice $\mathscr{B}$, and $mstable(P)$ be its multivalued stable model, as defined in [3], which is the lowest w.r.t. the information order. Then the semantics of P w.r.t. the default interpretation $\mathscr{D}$ coincides with $mstable(P)$.*

We show also that our semantics captures the α-fixed models of extended programs on bilattices, introduced by the author in [7]. For different logical values α, in particular false, true and $\perp$, the α-models provide various meanings to the same program, depending on how one chooses to complete the missing information by adopting a pessimistic, optimistic, or skeptical approach respectively. It was proven in [7] that α-fixed models capture successful conventional semantics as the well-founded semantics [10], the three-valued stable semantics [8], the bi-valued stable semantics [4] and the Kripke-Kleene semantics [1]. Thus the semantics for information integration presented in this work is a natural extension of the above successful conventional bi-valued or three-valued semantics of conventional logic programs.

Proposition 5. *Given an extended program P considered on the bilattice $\mathscr{B}$, for any logical value α from $\mathscr{B}$, the α-fixed model of P, as defined in [7], coincides with the semantics of P w.r.t. the default interpretation that uniformly assigns the value α to any ground atom, as defined in the current approach.*

We are currently studying how the semantics defined for information integration in this approach, strongly related to the multivalued stable models (that in turn generalize the conventional stable models), can be related to recent work, as for instance [6], which provides a logic programming based approach making use of a program semantics based on stable models, for merging belief bases. We currently investigate how the two approaches integrating information/beliefs and their corresponding program semantics can be compared, given the link with the stable model concept, although the present framework seems more general as based on multivalued logics.

References

[1] M. C. Fitting. A Kripke-Kleene semantics for logic programs. *J. of Logic Program.*, 2(4):295–312, 1985.

[2] M. C. Fitting. Bilattices and the semantics of logic programming. *J. of Logic Program.*, 11(1–2):91–116, 1991.

[3] M. C. Fitting. Fixpoint semantics for logic programming—a survey. *Theor. Comput. Sci.*, 278(1–2):25–51, 2002.

[4] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, eds., *Proc. of 5th Int. Conf. and Symp. on Logic Programming (Washington, DC, Aug. 1988)*, pp. 1070–1080. MIT Press, 1988.

[5] M. L. Ginsberg. Multivalued logics: a uniform approach to reasonning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[6] J. Hue, O. Papini, and E. Würbel. Merging belief bases represented by logic programs. In C. Sossai and G. Chemello, eds., *Proc. of 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2009 (Verona, July 2009)*, v. 5590 of *Lect. Notes in Artif. Intell.*, pp. 371–382. Springer, 2009.

[7] Y. Loyer, N. Spyratos, and D. Stamate. Parameterised semantics for logic programs—a unifying framework. *Theor. Comput. Sci.*, 308(1–3):429–447, 2003.

[8] T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundam. Inform.*, 13(4):445–463, 1990.

[9] D. Stamate. Information representation through extended logic programs in bilattices. In B. Bouchon-Meunier et al., eds., *Uncertainty and Intelligent Information Systems*, pp. 419–432. World Scientific, 2008.

[10] A. van Gelder, K. S. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. of ACM*, 38(3):620–650, 1991.

Fixed Points on Partial Randomness

Kohtaro Tadaki*
Research and Development Initiative, Chuo University
JST CREST
1–13–27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
tadaki@kc.chuo-u.ac.jp

Abstract

Algorithmic information theory (AIT, for short) is a theory of program-size and algorithmic randomness. One of the primary concepts of AIT is the *Kolmogorov complexity* $K(s)$ of a finite binary string s, which is defined as the length of the shortest binary program for a universal decoding algorithm to output s. In this paper, we report on a quite new type of fixed point in computer science, called a *fixed point on partial randomness*. In the research of AIT, it is important to consider the notion of the *compression rate* of a real T, which is defined as the real $\lim_{n\to\infty} K(T\restriction_n)/n$, where $T\restriction_n$ is the first n bits of the base-two expansion of T. The notion of the *partial randomness* of a real is a stronger representation of the compression rate. Our fixed point theorems on partial randomness give sufficient conditions for a real $T \in (0,1)$ to satisfy that the partial randomness of T equals to T and therefore the compression rate of T equals to T. The fixed point theorems are obtained in the framework of the statistical mechanical interpretation of AIT developed by our works [K. Tadaki, Local Proceedings of CiE 2008, pp. 425–434, 2008] and [K. Tadaki, Proceedings of LFCS'09, Springer's LNCS, vol. 5407, pp. 422–440, 2009]. As an original contribution of this paper, we present a simple and self-contained proof of the fixed point theorem on partial randomness.

1 Introduction and Summary

Algorithmic information theory (AIT, for short) is a framework to apply information-theoretic and probabilistic ideas to recursive function theory. One of the primary concepts of AIT is the *Kolmogorov complexity* (or *program-size complexity*) $K(s)$ of a finite binary string s, which is defined as the length of the shortest binary input for a universal decoding algorithm U, called an *optimal prefix-free machine*, to output s. By the definition, $K(s)$ is thought to represent the amount of randomness contained in s, which cannot be captured in a computational manner. In particular, the notion of Kolmogorov complexity plays a crucial role in characterizing the *randomness* of an infinite binary string, or equivalently, a real.

In this paper, we report on a quite new type of fixed point in computer science, called a *fixed point on partial randomness* [15, 16]. In the research of AIT, it is important to consider the notion of the *compression rate* of a real α, which is defined as the real $\lim_{n\to\infty} K(\alpha\restriction_n)/n$, where $\alpha\restriction_n$ is the first n bits of the base-two expansion of α. The notion of the *partial randomness* of a real is a stronger representation of the compression rate. *The fixed point theorems on partial randomness* give a sufficient condition for a real $T \in (0,1)$ to be a fixed point on partial randomness, i.e., to satisfy that the partial randomness of T equals to T.[1] One form of the fixed point theorems on partial randomness is presented as follows: For every $T \in (0,1)$, if $Z(T)$ is a computable real, then the partial randomness of T equals to T, and therefore the compression rate of T equals to T, i.e.,

$$\lim_{n\to\infty} \frac{K(T\restriction_n)}{n} = T, \tag{1}$$

*The work of the author was supported by KAKENHI, Grant-in-Aid for Scientific Research (C) (20540134), by SCOPE of the Ministry of Internal Affairs and Communications of Japan, and by CREST of the Japan Science and Technology Agency.

[1] The fixed point theorems on partial randomness were called fixed point theorems on compression rate in [15].

where $Z(T)$ is defined by

$$Z(T) := \sum_{U(p)\text{ is defined}} 2^{-\frac{|p|}{T}}. \tag{2}$$

Intuitively, we might interpret the meaning of (1) as follows: Consider imaginarily a file of infinite size whose content is

"The compression rate of this file is 0.100111001......"

When this file is compressed, the compression rate of this file actually equals to 0.100111001......, as the content of this file says. This situation is self-referential and forms a fixed point.

The fixed point theorems on partial randomness are obtained in the framework of the statistical mechanical interpretation of AIT developed by the works [15, 16]. In the development of the interpretation, we introduced the notion of *thermodynamic quantities at temperature T*, such as partition function $Z(T)$, free energy $F(T)$, energy $E(T)$, and statistical mechanical entropy $S(T)$, into AIT. These quantities are real functions of a real argument $T > 0$, and are defined based on the halting set of the optimal prefix-free machine U, like in (2). We proved that if T is a computable real with $0 < T < 1$ then, for each of the thermodynamic quantities at temperature T, the partial randomness of its value equals to T, and therefore the compression rate of its value equals to T. Thus, the temperature T plays a role as the partial randomness of all the thermodynamic quantities in the statistical mechanical interpretation of AIT. Furthermore, we showed that this situation holds for the temperature T itself, which is a thermodynamic quantity of itself in thermodynamics and statistical mechanics. Namely, we proved the fixed point theorems on partial randomness presented above, where the computability of $Z(T)$ gives a sufficient condition for $T \in (0,1)$ to be a fixed point on partial randomness. In addition, we showed that the computability of each of the remaining thermodynamic quantities $F(T)$, $E(T)$, and $S(T)$ also gives the sufficient condition. Moreover, based on the "statistical mechanical" relation $F(T) = -T\log_2 Z(T)$, we showed that the computability of $F(T)$ gives completely different fixed points from the computability of $Z(T)$.

The paper is organized as follows. We begin in Section 2 with some preliminaries to AIT and partial randomness. In Section 3, we give the definitions of the thermodynamic quantities in AIT and investigate their properties on partial randomness. The fixed point theorems on partial randomness are presented in Section 4. As an original contribution of this paper, in Section 5 we present a simple and self-contained proof of the fixed point theorem on partial randomness by $Z(T)$ stated above. We refer the reader to our original papers [15, 16] for the detail of this work and its related topics.

2 Preliminaries

We first review some basic notation and definitions which will be used in this paper. $\mathbb{N} = \{0,1,2,3,\dots\}$ is the set of natural numbers, and $\mathbb{N}^+$ is the set of positive integers. $\mathbb{Q}$ is the set of rationals, and $\mathbb{R}$ is the set of reals. $\{0,1\}^* = \{\lambda,0,1,00,01,10,11,000,\dots\}$ is the set of finite binary strings, where λ denotes the *empty string*. For any $s \in \{0,1\}^*$, $|s|$ is the *length* of s. A subset V of $\{0,1\}^*$ is called *prefix-free* if no string in V is a prefix of another string in V. It is easy to show that every prefix-free set $V \subset \{0,1\}^*$ satisfies the so-called *Kraft inequality* $\sum_{s\in V} 2^{-|s|} \le 1$. For any partial function f, the domain of definition of f is denoted by $\operatorname{dom} f$. We write "r.e." instead of "recursively enumerable." A real α is called *computable* if there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $|\alpha - f(n)| < 1/n$ for all $n \in \mathbb{N}^+$. For any $\alpha \in \mathbb{R}$ and $n \in \mathbb{N}^+$, we denote by $\alpha{\restriction}_n \in \{0,1\}^*$ the first n bits of the base-two expansion of $\alpha - \lfloor \alpha \rfloor$ with infinitely many zeros, where $\lfloor \alpha \rfloor$ is the greatest integer less than or equal to α. For example, in the case of $\alpha = 5/8$, $\alpha{\restriction}_6 = 101000$.

In what follows we concisely review some definitions and results of AIT. For the detail, see [3, 4, 8, 5]. A *prefix-free machine* is a partial recursive function $M\colon \{0,1\}^* \to \{0,1\}^*$ such that $\operatorname{dom} M$ is a prefix-free set. For any prefix-free machine M and any $s \in \{0,1\}^*$, $K_M(s)$ is defined by $K_M(s) = \min\{\,|p| \mid p \in \{0,1\}^* \;\&\; M(p) = s\,\}$ (may be ∞). A prefix-free machine U is said to be *optimal* if for each prefix-free machine M there exists $c \in \mathbb{N}$ with the following property; if $p \in \operatorname{dom} M$, then there is $q \in \operatorname{dom} U$ for which $U(q) = M(p)$ and $|q| \le |p| + c$. There exists an optimal prefix-free machine. We choose a particular optimal prefix-free machine U as the standard one for use, and define $K(s)$ as $K_U(s)$, which is referred to as the *Kolmogorov complexity* of s or the *program-size complexity* of s. It follows that for every prefix-free machine M there exists $c \in \mathbb{N}$ such that, for every $s \in \{0,1\}^*$,

$$K(s) \le K_M(s) + c. \tag{3}$$

Based on this we can show that there exists $c \in \mathbb{N}$ such that, for every $s \neq \lambda$,

$$K(s) \le |s| + 2\log_2 |s| + c. \tag{4}$$

Using (3), it is also easy to show that, for every partial recursive function $\Psi\colon \{0,1\}^* \to \{0,1\}^*$, there exists $c \in \mathbb{N}$ such that, for every $s \in \operatorname{dom}\Psi$,

$$K(\Psi(s)) \le K(s) + c. \tag{5}$$

An element of $\operatorname{dom} U$ is called a *program for* U.

Let T be an arbitrary real with $0 < T \le 1$. In the work [14], we introduced several notions of the partial randomness of a real by parameterizing the notions of randomness of a real by a real T, as follows. Let $\alpha \in \mathbb{R}$. We say that α is *weakly Chaitin T-random* if there exists $c \in \mathbb{N}$ such that $Tn - c \le K(\alpha\restriction_n)$ for all $n \in \mathbb{N}^+$. On the other hand, we say that α is *T-compressible* if $K(\alpha\restriction_n) \le Tn + o(n)$, which is equivalent to $\limsup_{n\to\infty} K(\alpha\restriction_n)/n \le T$. Thus, if α is weakly Chaitin T-random and T-compressible, then

$$\lim_{n\to\infty} \frac{K(\alpha\restriction_n)}{n} = T. \tag{6}$$

The left-hand side of (6) is referred to as the *compression rate* of a real α in general. Note, however, that (6) does not necessarily imply that α is weakly Chaitin T-random. Thus, the notion of partial randomness is a stronger representation of the compression rate. We say that α is *Chaitin T-random* if $\lim_{n\to\infty} K(\alpha\restriction_n) - Tn = \infty$. Obviously, if α is Chaitin T-random, then α is weakly Chaitin T-random. However, in 2005 Reimann and Stephan [9] showed that, in the case of $T < 1$, the converse does not necessarily hold.

3 Thermodynamic Quantities in AIT

We introduce the notion of thermodynamic quantities into AIT in the following manner.

In statistical mechanics, the partition function $Z_{\rm sm}(T)$, free energy $F_{\rm sm}(T)$, energy $E_{\rm sm}(T)$, and entropy $S_{\rm sm}(T)$ at temperature T are given as follows:

$$\begin{aligned} Z_{\rm sm}(T) &= \sum_{x\in X} e^{-\frac{E_x}{k_{\rm B}T}}, & F_{\rm sm}(T) &= -k_{\rm B}T \ln Z_{\rm sm}(T), \\ E_{\rm sm}(T) &= \frac{1}{Z_{\rm sm}(T)} \sum_{x\in X} E_x e^{-\frac{E_x}{k_{\rm B}T}}, & S_{\rm sm}(T) &= \frac{E_{\rm sm}(T) - F_{\rm sm}(T)}{T}, \end{aligned} \tag{7}$$

where X is a complete set of energy eigenstates of a quantum system and E_x is the energy of an energy eigenstate x. The constant $k_{\rm B}$ is called *the Boltzmann Constant*, and the ln denotes the natural logarithm.[2]

[2] For the thermodynamic quantities in statistical mechanics, see e.g. Chapter 16 of [1]. To be precise, the partition function is not a thermodynamic quantity but a statistical mechanical quantity.

We introduce the notion of thermodynamic quantities into AIT by performing Replacements 1 below for the thermodynamic quantities (7) in statistical mechanics.

Replacements 1.

(i) Replace the complete set X of energy eigenstates x by the set $\operatorname{dom} U$ *of all programs p for U.*

(ii) Replace the energy E_x of an energy eigenstate x by the length $|p|$ of a program p.

(iii) Set the Boltzmann Constant k_{B} to $1/\ln 2$. □

For that purpose, we first choose a particular recursive enumeration $p_1, p_2, p_3, p_4, \dots$ of the infinite r.e. set $\operatorname{dom} U$ as the standard one for use throughout the rest of this paper.[3] Then, motivated by the formulae (7) and taking into account Replacements 1, we introduce the notion of thermodynamic quantities into AIT as follows.

Definition 3.1 (thermodynamic quantities in AIT, [15])**.** *Let T be any real with $T > 0$.*

(i) The partition function $Z(T)$ at temperature T is defined as $\lim_{k\to\infty} Z_k(T)$ *where*

$$Z_k(T) = \sum_{i=1}^{k} 2^{-\frac{|p_i|}{T}}.$$

(ii) The free energy $F(T)$ at temperature T is defined as $\lim_{k\to\infty} F_k(T)$ *where*

$$F_k(T) = -T \log_2 Z_k(T).$$

(iii) The energy $E(T)$ at temperature T is defined as $\lim_{k\to\infty} E_k(T)$ *where*

$$E_k(T) = \frac{1}{Z_k(T)} \sum_{i=1}^{k} |p_i| 2^{-\frac{|p_i|}{T}}.$$

(iv) The statistical mechanical entropy $S(T)$ at temperature T is defined as $\lim_{k\to\infty} S_k(T)$ *where*

$$S_k(T) = \frac{E_k(T) - F_k(T)}{T}.$$ □

Since $\operatorname{dom} U$ is prefix-free, we first see that the Kraft inequality $Z(1) \le 1$ holds. The real $Z(1)$ is precisely a Chaitin Ω number introduced by Chaitin [3]. For every $T \in (0,1]$, $Z(T)$ converges and $Z(T) \le 1$ since $2^{-|p_i|/T} \le 2^{-|p_i|}$.

Theorem 3.2 (properties of $Z(T)$ and $F(T)$, [14, 15])**.** *If T is a computable real with* $0 < T \le 1$, *then each of $Z(T)$ and $F(T)$ converges and is weakly Chaitin T-random and T-compressible.* □

Theorem 3.3 (properties of $E(T)$ and $S(T)$, [15])**.** *If T is a computable real with* $0 < T < 1$, *then each of $E(T)$ and $S(T)$ converges and is Chaitin T-random and T-compressible.* □

The above two theorems show that if T is a computable real with $T \in (0,1)$ then the temperature T equals to the partial randomness (and therefore the compression rate) of the values of all the thermodynamic quantities in Definition 3.1. Note that the weak Chaitin T-randomness in Theorems 3.2 is strengthen to the Chaitin T-randomness in Theorems 3.3.

[3] Actually, the enumeration $\{p_i\}$ can be chosen quite arbitrarily, and the results of this paper are independent of the choice of $\{p_i\}$. This is because the sum $\sum_{i=1}^{k} 2^{-|p_i|/T}$ and $\sum_{i=1}^{k} |p_i| 2^{-|p_i|/T}$ in Definition 3.1 are positive term series and converge as $k \to \infty$ for every $T \in (0,1)$.

4 Fixed Point Theorems on Partial Randomness

In statistical mechanics or thermodynamics, among all thermodynamic quantities one of the most typical thermodynamic quantities is temperature itself. Inspired by this fact in physics and by the observation in the previous section that the temperature T equals to the partial randomness of the values of the thermodynamic quantities in the statistical mechanical interpretation of AIT, the following question arises naturally: Can the partial randomness of the temperature T equal to the temperature T itself in the statistical mechanical interpretation of AIT ? This question is rather self-referential. However, we can answer it affirmatively in the following form.

Theorem 4.1 (fixed point theorem by $Z(T)$, [15]). *For every $T \in (0,1)$, if $Z(T)$ is computable, then T is weakly Chaitin T-random and T-compressible, and therefore $\lim_{n\to\infty} K(T\restriction_n)/n = T$.* □

Theorem 4.1 is just a *fixed point theorem on partial randomness*, where the computability of the value $Z(T)$ gives a sufficient condition for a real $T \in (0,1)$ to be a fixed point on partial randomness. Thus, the above observation that the temperature T equals to the partial randomness of the values of the thermodynamic quantities in the statistical mechanical interpretation of AIT is further confirmed. In addition, we can show that fixed point theorems of the same form as Theorem 4.1 hold also for the free energy $F(T)$, energy $E(T)$, and statistical mechanical entropy $S(T)$, as follows. Thus we confirm the above observation much further.

Theorem 4.2 (fixed point theorem by $F(T)$, [16]). *For every $T \in (0,1)$, if $F(T)$ is computable then T is weakly Chaitin T-random and T-compressible.* □

Theorem 4.3 (fixed point theorem by $E(T)$, [16]). *For every $T \in (0,1)$, if $E(T)$ is computable then T is Chaitin T-random and T-compressible.* □

Theorem 4.4 (fixed point theorem by $S(T)$, [16]). *For every $T \in (0,1)$, if $S(T)$ is computable then T is Chaitin T-random and T-compressible.* □

Note that the weak Chaitin T-randomness of T in Theorems 4.1 is strengthen to the Chaitin T-randomness of T in Theorems 4.3 and 4.4. Theorem 4.1 will be proved in Section 5 in a self-contained manner. Since the function $Z(T)$ of T is monotonically increasing and continuous on $(0,1)$, and the set of all computable reals is dense in $\mathbb{R}$, the following theorem holds for the sufficient condition of Theorem 4.1. The exactly same theorem holds for each of $F(T)$, $E(T)$, and $S(T)$ also [16].

Theorem 4.5 ([15]). *The set $\{T \in (0,1) \mid Z(T)$ is computable$\}$ is dense in $(0,1)$.* □

Using the "statistical mechanical" relation $F(T) = -T\log_2 Z(T)$ we can show Theorem 4.6 below. Thus, the computability of $F(T)$ gives completely different fixed points from the computability of $Z(T)$. This implies that neither the computability of $Z(T)$ nor the computability of $F(T)$ is the necessary condition for $T \in (0,1)$ to be a fixed point on partial randomness at all.

Theorem 4.6 ([16]). *There does not exist $T \in (0,1)$ such that both $Z(T)$ and $F(T)$ are computable.* □

Using the property of T as a fixed point in Theorems 4.1, we can show the following.

Theorem 4.7 ([16]). *$S_a \cap S_b = \emptyset$ for any distinct computable reals $a,b \in (0,1]$, where $S_a = \{T \in (0,1) \mid Z(aT)$ is computable$\}$.*

Proof. Let $T \in (0,1)$, and let a be a computable real with $a \in (0,1]$. Suppose that $Z(aT)$ is computable. Then, by Theorem 4.1, $\lim_{n\to\infty} K((aT)\restriction_n)/n = aT$. Since $K((aT)\restriction_n) = K(T\restriction_n) + O(1)$ it follows that $\lim_{n\to\infty} K(T\restriction_n)/n = aT$. Thus, for every computable reals $a,b \in (0,1]$, if $S_a \cap S_b \neq \emptyset$ then $a = b$. □

As a corollary of Theorem 4.7, we have the following, for example.

Corollary 4.8 ([16]). *For every $T \in (0,1)$, if $Z(T)$ is computable, then $Z(T/n)$ is not computable for every $n \in \mathbb{N}^+$ with $n \geq 2$. In other words, for every $T \in (0,1)$, if the sum $\sum_{i=1}^{\infty} 2^{-|p_i|/T}$ is computable, then the corresponding power sum $\sum_{i=1}^{\infty} \left(2^{-|p_i|/T}\right)^n$ is not computable for every $n \in \mathbb{N}^+$ with $n \geq 2$.* □

5 The Proof of Theorem 4.1

As an original contribution of this paper, we present a simple and self-contained proof of Theorem 4.1 in what follows. We first recall the notion of right computable enumerability and left computable enumerability of a real. A real α is called *right computably enumerable* (*right-c.e.*, for short) if there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $\alpha \leq f(n)$ for all $n \in \mathbb{N}^+$ and $\lim_{n\to\infty} f(n) = \alpha$. Right-c.e. reals are also called *right-computable*. On the other hand, a real α is called *left computably enumerable* (*left-c.e.*, for short) if $-\alpha$ is right-c.e. Left-c.e. reals are also called *left-computable*. It is then easy to show the following theorem.

Theorem 5.1. *Let $\alpha \in \mathbb{R}$.*

(i) α is computable if and only if α is both right-c.e. and left-c.e.

(ii) α is right-c.e. if and only if the set $\{\, r \in \mathbb{Q} \mid \alpha < r \,\}$ is r.e. □

Theorem 4.1 follows immediately from Theorem 5.2, Theorem 5.3, and Theorem 5.4 below, as well as from Theorem 5.1 (i).

Theorem 5.2. *For every $T \in (0,1)$, if $Z(T)$ is right-c.e. then T is weakly Chaitin T-random.*

Proof. First, for each $k \in \mathbb{N}^+$ and each real $x > 0$, we define $W_k(x)$ as $\sum_{i=1}^{k} |p_i| 2^{-|p_i|/x}$. We show that, for each $x \in (0,1)$, $W_k(x)$ converges as $k \to \infty$. Let x be an arbitrary real with $x \in (0,1)$. Since $x < 1$, there is $l_0 \in \mathbb{N}^+$ such that $(\log_2 l)/l \leq 1/x - 1$ for all $l \geq l_0$. Then there is $k_0 \in \mathbb{N}^+$ such that $|p_i| \geq l_0$ for all $i > k_0$. Thus, we see that, for each $i > k_0$,

$$|p_i| 2^{-\frac{|p_i|}{x}} = 2^{-(\frac{1}{x} - \frac{\log_2 |p_i|}{|p_i|})|p_i|} \leq 2^{-|p_i|}.$$

Hence, for each $k > k_0$, $W_k(x) - W_{k_0}(x) = \sum_{i=k_0+1}^{k} |p_i| 2^{-|p_i|/x} \leq \sum_{i=k_0+1}^{k} 2^{-|p_i|} < Z(1)$. Therefore, since $\{W_k(x)\}_k$ is an increasing sequence of reals bounded to the above, it converges as $k \to \infty$, as desired. For each $x \in (0,1)$, we define a positive real number $W(x)$ as $\lim_{k\to\infty} W_k(x)$.

On the other hand, since $Z(T)$ is right-c.e. by the assumption, there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $Z(T) \leq f(m)$ for all $m \in \mathbb{N}^+$, and $\lim_{m\to\infty} f(m) = Z(T)$.

We choose a particular real t with $T < t < 1$. Then, for each $i \in \mathbb{N}^+$, using the mean value theorem we see that

$$2^{-\frac{|p_i|}{x}} - 2^{-\frac{|p_i|}{T}} < \frac{\ln 2}{T^2} |p_i| 2^{-\frac{|p_i|}{t}} (x - T)$$

for all $x \in (T,t)$. We then choose a particular $c \in \mathbb{N}$ with $W(t) \ln 2/T^2 \leq 2^c$. Here, the limit value $W(t)$ exists, since $0 < t < 1$. It follows that

$$Z_k(x) - Z_k(T) < 2^c (x - T) \tag{8}$$

for all $k \in \mathbb{N}^+$ and $x \in (T,t)$. We also choose a particular $n_0 \in \mathbb{N}^+$ such that $0.(T{\restriction_n}) + 2^{-n} < t$ for all $n \geq n_0$. Such n_0 exists since $T < t$ and $\lim_{n\to\infty} 0.(T{\restriction_n}) + 2^{-n} = T$. Since $T{\restriction_n}$ is the first n bits of the base-two expansion of T with infinitely many zeros, we then see that $T < 0.(T{\restriction_n}) + 2^{-n} < t$ for all $n \geq n_0$.

Now, given $T\restriction_n$ with $n \geq n_0$, one can find $k_0, m_0 \in \mathbb{N}^+$ such that $f(m_0) < Z_{k_0}(0.(T\restriction_n)+2^{-n})$. This is possible from $Z(T) < Z(0.(T\restriction_n)+2^{-n})$, $\lim_{k\to\infty} Z_k(0.(T\restriction_n)+2^{-n}) = Z(0.(T\restriction_n)+2^{-n})$, and the properties of f. It follows from $Z(T) \leq f(m_0)$ and (8) that $\sum_{i=k_0+1}^{\infty} 2^{-|p_i|/T} = Z(T) - Z_{k_0}(T) < Z_{k_0}(0.(T\restriction_n)+2^{-n}) - Z_{k_0}(T) < 2^{c-n}$. Hence, for every $i > k_0$, $2^{-|p_i|/T} < 2^{c-n}$ and therefore $Tn - Tc < |p_i|$. Thus, by calculating the set $\{\, U(p_i) \mid i \leq k_0 \,\}$ and picking any one finite binary string which is not in this set, one can then obtain an $s \in \{0,1\}^*$ such that $Tn - Tc < K(s)$.

Hence, there exists a partial recursive function $\Psi\colon \{0,1\}^* \to \{0,1\}^*$ such that $Tn - Tc < K(\Psi(T\restriction_n))$ for all $n \geq n_0$. Using (5), there is $c_\Psi \in \mathbb{N}$ such that $K(\Psi(T\restriction_n)) \leq K(T\restriction_n) + c_\Psi$ for all $n \geq n_0$. Therefore, $Tn - Tc - c_\Psi < K(T\restriction_n)$ for all $n \geq n_0$. It follows that T is weakly Chaitin T-random. □

Theorem 5.3. *For every $T \in (0,1)$, if $Z(T)$ is right-c.e., then T is also right-c.e.*

Proof. Since $Z(T)$ is right-c.e., there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $Z(T) \leq f(m)$ for all $m \in \mathbb{N}^+$, and $\lim_{m\to\infty} f(m) = Z(T)$. Thus, since $Z(x)$ is an increasing function of $x \in (0,1]$, we see that, for every $x \in \mathbb{Q}$ with $0 < x < 1$, $T < x$ if and only if there are $m, k \in \mathbb{N}^+$ such that $f(m) < Z_k(x)$. It follows from Theorem 5.1 (ii) that T is right-c.e. □

Theorem 5.4. *For every $T \in (0,1)$, if $Z(T)$ is left-c.e. and T is right-c.e., then T is T-compressible.*

Proof. For each $i \in \mathbb{N}^+$, using the mean value theorem we see that

$$2^{-\frac{|p_1|}{t}} - 2^{-\frac{|p_1|}{T}} > (\ln 2)\,|p_1|\,2^{-\frac{|p_1|}{T}}(t-T)$$

for all $t \in (T,1)$. We choose a particular $c \in \mathbb{N}^+$ such that $(\ln 2)\,|p_1|\,2^{-\frac{|p_1|}{T}} \geq 2^{-c}$. Then, it follows that

$$Z_k(t) - Z_k(T) > 2^{-c}(t-T) \tag{9}$$

for all $k \in \mathbb{N}^+$ and $t \in (T,1)$.

Since T is a right-c.e. real with $T < 1$ by the assumption, there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $T < f(l) < 1$ for all $l \in \mathbb{N}^+$, and $\lim_{l\to\infty} f(l) = T$. On the other hand, since $Z(T)$ is left-c.e. by the assumption, there exists a total recursive function $g\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $g(m) \leq Z(T)$ for all $m \in \mathbb{N}^+$, and $\lim_{m\to\infty} g(m) = Z(T)$. Let $\Omega = Z(1)$. By Theorem 3.2, $Z(1)$ is weakly Chaitin 1-random and therefore $Z(1) \notin \mathbb{Q}$. Thus, the base-two expansion of Ω is unique and contains infinitely many ones, and $0 < \Omega < 1$ in particular.

Given n and $\Omega\restriction_{\lceil Tn\rceil}$ (i.e., the first $\lceil Tn\rceil$ bits of the base-two expansion of Ω), one can find $k_0 \in \mathbb{N}^+$ such that $0.(\Omega\restriction_{\lceil Tn\rceil}) < \sum_{i=1}^{k_0} 2^{-|p_i|}$. This is possible since $0.(\Omega\restriction_{\lceil Tn\rceil}) < \Omega$ and $\lim_{k\to\infty} \sum_{i=1}^{k} 2^{-|p_i|} = \Omega$. It is then easy to see that $\sum_{i=k_0+1}^{\infty} 2^{-|p_i|} = \Omega - \sum_{i=1}^{k_0} 2^{-|p_i|} < 2^{-\lceil Tn\rceil} \leq 2^{-Tn}$. Using the inequality $a^d + b^d \leq (a+b)^d$ for any reals $a, b > 0$ and $d \geq 1$, it follows that

$$Z(T) - Z_{k_0}(T) = \sum_{i=k_0+1}^{\infty} 2^{-\frac{|p_i|}{T}} < 2^{-n}. \tag{10}$$

Note that $\lim_{l\to\infty} Z_{k_0}(f(l)) = Z_{k_0}(T)$. Thus, since $Z_{k_0}(T) < Z(T)$, one can then find $l_0, m_0 \in \mathbb{N}^+$ such that $Z_{k_0}(f(l_0)) < g(m_0)$. It follows from (10) and (9) that $2^{-n} > g(m_0) - Z_{k_0}(T) > Z_{k_0}(f(l_0)) - Z_{k_0}(T) > 2^{-c}(f(l_0) - T)$. Thus, $0 < f(l_0) - T < 2^{c-n}$. Let t_n be the first n bits of the base-two expansion of the rational number $f(l_0)$ with infinitely many zeros. Then, $|\,f(l_0) - 0.t_n\,| < 2^{-n}$. It follows from $|\,T - 0.(T\restriction_n)\,| < 2^{-n}$ that $|\,0.(T\restriction_n) - 0.t_n\,| < (2^c+2)2^{-n}$. Hence, $T\restriction_n = t_n, t_n \pm 1, t_n \pm 2, \dots, t_n \pm (2^c+1)$, where $T\restriction_n$ and t_n are regarded as a dyadic integer. Thus, there are still $2^{c+1}+3$ possibilities of $T\restriction_n$, so that one needs only $c+2$ bits more in order to determine $T\restriction_n$.

Thus, there exists a partial recursive function $\Phi\colon \mathbb{N}^+ \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that

$$\forall n \in \mathbb{N}^+ \quad \exists s \in \{0,1\}^* \quad |s| = c+2 \ \&\ \Phi(n, \Omega\restriction_{\lceil Tn\rceil}, s) = T\restriction_n\,.$$

It follows from (4) that $K(T\restriction_n) \le |\Omega\restriction_{\lceil Tn \rceil}| + o(n) \le Tn + o(n)$, which implies that T is T-compressible. □

6 Concluding Remarks

By a series of works of Ryabko [10, 11], Staiger [12, 13], Lutz [6], Tadaki [14], and Mayordomo [7] over the last two decades, the equivalence between the notion of *Hausdorff dimension* and the notion of compression rate by Kolmogorov complexity seems to be established at present. In particular, Tadaki [14] considered the equivalence between the notion of Hausdorff dimension and the notion of partial randomness as well as the compression rate. In the context of the subject of the equivalence, we can consider the notion of the *dimension* of an individual real in particular, and this notion plays a crucial role in the subject. Our fixed point theorems on partial randomness give sufficient conditions for a real $T \in (0,1)$ to satisfy that the dimension of T equals to T. Thus, it would be interesting if we could develop the subject of the equivalence further in a new direction based on the notion of fixed point on partial randomness.

References

[1] H. B. Callen. *Thermodynamics and an Introduction to Thermostatistics*, 2nd ed. John Wiley & Sons, 1985.

[2] C. S. Calude and M. A. Stay. Natural halting probabilities, partial randomness, and zeta functions. *Inform. and Comput.*, 204(11):1718–1739, 2006.

[3] G. J. Chaitin. A theory of program size formally identical to information theory. *J. of ACM*, 22(3):329–340, 1975.

[4] G. J. Chaitin. *Algorithmic Information Theory*. Cambridge Univ. Press, 1987.

[5] R. G. Downey and D. R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer, to appear.

[6] J. H. Lutz. Gales and the constructive dimension of individual sequences. In U. Montanari et al., eds., *Proc. of 27th Int. Coll. on Automata, Languages and Programming, ICALP 2000 (Geneva, July 2000)*, v. 1853 of *Lect. Notes in Comput. Sci.*, pp. 902–913. Springer, 2000.

[7] E. Mayordomo. A Kolmogorov complexity characterization of constructive Hausdorff dimension. *Inform. Process. Lett.*, 84(1):1–3, 2002.

[8] A. Nies. *Computability and Randomness*. Oxford Univ. Press, 2009.

[9] J. Reimann and F. Stephan. On hierarchies of randomness tests. In S. S. Goncharov et al., eds., *Proc. of 9th Asian Logic Conference (Novosibirsk, Aug. 2005)*, pp. 215–232. World Scientific, 2006.

[10] B. Ya. Ryabko. Coding of combinatorial sources and Hausdorff dimension. *Soviet Math. Dokl.*, 30:219–222, 1984.

[11] B. Ya. Ryabko. Noiseless coding of combinatorial sources, Hausdorff dimension, and Kolmogorov complexity. *Probl. of Inform. Transm.*, 22:170–179, 1986.

[12] L. Staiger. Kolmogorov complexity and Hausdorff dimension. *Inform. and Comput.*, 103(2):159–194, 1993.

[13] L. Staiger. A tight upper bound on Kolmogorov complexity and uniformly optimal prediction. *Theory of Comput. Syst.*, 31(3):215–229, 1998.

[14] K. Tadaki. A generalization of Chaitin's halting probability Ω and halting self-similar sets. *Hokkaido Math. J.*, 31(1):219–253, 2002.

[15] K. Tadaki. A statistical mechanical interpretation of algorithmic information theory. In *Local Proceedings of Computability in Europe 2008, CiE 2008 (Athens, June 2008)*, pp. 425–443. Univ. of Athens, 2008. Extended version: `http://arxiv.org/abs/0801.4194v1`

[16] K. Tadaki. Fixed point theorems on partial randomness. In S. Artemov and A. Nerode, eds., *Proc. of Symp. on Logical Foundations of Computer Science 2009, LFCS 2009 (Deerfield Beach, FL, Jan. 2009)*, v. 5407 of *Lect. Notes in Comput. Sci.*, pp. 422-440. Springer, 2009. Extended version: `http://arxiv.org/abs/0903.3433`

Fixed-Point Computations over Functions on Integers with Operations Min, Max and Plus

Yoshinori Tanabe and Masami Hagiya
Dept. of Computer Science, Graduate School of Inform. Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
tanabe@ci.i.u-tokyo.ac.jp and hagiya@is.s.u-tokyo.ac.jp

Abstract

Various kinds of graph problems, including shortest path computation, proof-number search, dataflow analysis, etc., can be solved by fixed-point computations over functions defined on natural numbers or integers. In this paper, we prove that fixed-point computations are possible for the algebra $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty, -\infty\}$, which has the operators min, max and plus. Since $\mathbb{Z}_\infty$ is not well-ordered, we formulate a kind of acceleration technique to guarantee termination of fixed-point computations.

1 Introduction

Fixed-point computations on various algebraic structures are required in many fields of computer science. The simplest example of such an algebraic structure is the boolean algebra, $\mathbf{2} = \{0,1\}$, on which ordinary boolean operations are defined. Model-checking problems on Kripke structures [3] are reduced to fixed-point computations over functions from the set of states to **2**, where each parameter of a function corresponds to the value of a propositional variable at a state of the target Kripke structure, which relates the parameters belonging to one state with those belonging to adjacent states. The μ and ν operators in the modal μ-calculus [6] correspond to the least and greatest fixed-points of a system of such boolean functions.

Model-checking problems can be easily generalized by adopting algebraic structures other than **2**. Propositional variables are generalized to variables having value of the adopted algebraic structure, and boolean operations are replaced with operations on the algebraic structure. The modal μ-calculus can still be used as a language for expressing fixed-points, if the operators of the calculus are interpreted as operations on the algebraic structure. In fact, in our previous work [4], we adopted $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$, the set of natural numbers augmented with infinity, as an algebraic structure, and interpreted the operators $\vee$ and $\wedge$ as min and plus, respectively, over $\mathbb{N}_\infty$. The modal operators $\Diamond$ and $\Box$ were also interpreted accordingly. This kind of algebraic structure having min and plus is well known as min-plus algebra [8, 2]. As the values 0 and ∞ in $\mathbb{N}_\infty$ naturally correspond to 1 (true) and 0 (false) in **2**, respectively, we interpreted the μ and ν operators as the greatest and least fixed-points. We then formulated algorithms for computing fixed-points over generalized Kripke structures.

Interestingly, various kinds of graph problems can be expressed in the above framework, including shortest path computation, proof-number search [1], dataflow analysis, etc. For example, we can extend the approach taken by Lacey et al., who used CTL, a sublogic of the modal μ-calculus, to express complex conditions on a control flow graph for the purpose of program transformation [7]. A control flow graph is regarded as a Kripke structure. We introduce propositional symbols **accessx** and **updatex**, which hold at a node of the control flow graph if the program variable **x** is accessed and updated on the node, respectively. Then we can construct a formula that expresses, for example, the minimum number of accesses to the variable **x** after each update of **x**.

For applications as the above one, it is natural to introduce the max operator in addition to min and plus. In the above example, it becomes possible to express the maximum number of accesses to a program variable. As this paper shows, however, fixed-point computations become more involved if max is introduced.

Even in our previous work, fixed-point computations were not trivial. Since $\mathbb{N}_\infty$ is well-ordered, the greatest fixed-point can be calculated in a natural way — starting from ∞ and repeat calculating the next value to obtain a decreasing chain of values. However, this simple strategy cannot be applied to compute the least fixed-point. In our previous work, we developed a kind of acceleration technique to guarantee termination.

In this paper, by carefully extending the acceleration technique, we show that fixed-point computations are also possible for the algebra $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty, -\infty\}$, which has the operators min, max and plus. We formulate an algorithm that computes fixed-points of functions defined on $\mathbb{Z}_\infty$. Our previous work is subsumed by embedding $\mathbb{N}_\infty$ into $\mathbb{Z}_\infty$. The efficiency of fixed-point computations is also improved. In some cases, the new algorithm is more efficient than the old one.

2 Target Functions

We define the set $\mathscr{F}$ of functions that our algorithm targets. Roughly speaking, an element of $\mathscr{F}$ is a function on finite power of $\mathbb{Z}_\infty$, composed of operations min, max, plus, minus, and fixed-points. Since we allow the fixed-point operations, the functions need to be monotone with respect to parameters over which the fixed-point is calculated. Therefore, we need to keep track of *positive* and *negative* parameters. Another small issue is value of operations when operands are ∞ or $-\infty$. While most of them can be defined naturally, some decision on the value of $\infty + (-\infty)$ is needed. We introduce two different operators $+_\uparrow$ and $+_\downarrow$, and define $\infty +_\uparrow (-\infty) = (-\infty) +_\uparrow \infty = \infty$, and $\infty +_\downarrow (-\infty) = (-\infty) +_\downarrow \infty = -\infty$. If $\{x, y\} \neq \{\infty, -\infty\}$, then $x +_\uparrow y = x +_\downarrow y = x + y$.

We introduce some notations. Let $n, m, k \in \mathbb{N}$. The set $\{n+1, n+2, \ldots, n+m\}$ is denoted by $I_{n,m}$, and $I_{0,n}$ is denoted by I_n. The constant function on $\mathbb{Z}_\infty{}^n$ that takes value $c \in \mathbb{Z}_\infty$ is denoted by γ_c^n. The projection function on $\mathbb{Z}_\infty{}^n$ to $T = \{t_1, \ldots, t_k\} \subseteq I_n$ is denoted by π_T^n, i.e., $\pi_T^n : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^k$, $\pi_T^n(x_1, \ldots, x_n) = (x_{t_1}, \ldots, x_{t_k})$. We write $\pi_{\{m\}}^n$ as π_m^n. The superscript n of γ and π is often omitted if no confusion occurs. For function f, function $\pi_T f$ is defined by $(\pi_T f)(x) = \pi_T(f(x))$. If $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_k)$, then $(x_1, \ldots, x_n, y_1, \ldots, y_k)$ is denoted by (x, y). We intentionally abuse this notation: if (T, S) is a partition of I_n, $y = \pi_T x$, and $z = \pi_S x$, then we write (y, z) to express x, if T and S are clear from the context. For functions $f : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^m$ and $g : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^k$, $(f, g) : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^{m+k}$ is defined by $(f, g)(x) = (f(x), g(x))$. For $c \in \mathbb{Z}_\infty$, the n-tuple of c, i.e., $x \in \mathbb{Z}_\infty{}^n$ such that $\pi_i x = c$ for all $i \in I_n$, is also denoted by c.

For $n, k \in \mathbb{N}$ and $c \in \mathbb{Z}_\infty$, let $z_{k,c}^n \in \mathbb{Z}_\infty{}^n$ be defined by $\pi_k z_{k,c}^n = c$ and $\pi_i z_{k,c}^n = 0$ for all $i \in I_n \setminus \{k\}$.

For $x, y \in \mathbb{Z}_\infty{}^n$, we write $x \leq y$ if $\pi_i x \leq \pi_i y$ for all $i \in I_n$. If $x \leq y$ and $x \neq y$, we write $x < y$. If $\pi_i x < \pi_i y$ for all $i \in I_n$, we write $x \ll y$. If (T, S) is a partition of I_n, $x \leq y$, and $\pi_S x = \pi_S y$, we write $x \leq_T y$. A function $f : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^m$ is *monotone* (or *monotone increasing*) w.r.t. T if $x \leq_T y$ implies $f(x) \leq f(y)$, and *monotone decreasing* w.r.t. T if $x \leq_T y$ implies $f(x) \geq f(y)$.

For $f : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^n$, the k-th repetition of f is denoted by $f^{(k)}$. I.e., $f^{(k)} : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^n$, $f^{(0)}(x) = x$, and $f^{(k+1)}(x) = f(f^{(k)}(x))$.

We define the set $\mathscr{F}$ as the least set that satisfies the following conditions, together with the set $P(f)$ and $N(f)$ of positive and negative parameter indices of $f \in \mathscr{F}$, respectively.

- $\gamma_c \in \mathscr{F}$ and $P(\gamma_c) = N(\gamma_c) = \varnothing$.
- $\pi_T \in \mathscr{F}$, $P(\pi_T) = T$, and $N(\pi_T) = \varnothing$.
- If $f \in \mathscr{F}$, then $-f \in \mathscr{F}$, $P(-f) = N(f)$, and $N(-f) = P(f)$.
- If $f, g \in \mathscr{F}$, $P = P(f) \cup P(g)$, $N = N(f) \cup N(g)$, and $P \cap N = \varnothing$, then $h = (f, g)$, $f +_\uparrow g$, $f +_\downarrow g$, $\min(f, g)$, $\max(f, g) \in \mathscr{F}$, $P(h) = P$, and $N(h) = N$.

- If $f : {\mathbb{Z}_\infty}^{n+m} \to {\mathbb{Z}_\infty}^m$, $f \in \mathscr{F}$, and $I_{n,m} \cap N(f) = \varnothing$, then $h = \mathrm{LFP}(f), \mathrm{GFP}(f) \in \mathscr{F}$, $P(h) = P(f) \setminus I_{n,m}$, and $N(h) = N(f)$, where $\mathrm{LFP}(f) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ is defined so that for any $x \in {\mathbb{Z}_\infty}^n$, $\mathrm{LFP}(f)(x)$ is the least $y \in {\mathbb{Z}_\infty}^m$ such that $f(x,y) = y$. $\mathrm{GFP}(f)$ is defined similarly as the largest such y.

The existence of LFP and GFP in the definition can be proved by simultaneous induction with the fact that for all $f \in \mathscr{F}$, f is monotone increasing w.r.t. $P(f)$ and monotone decreasing w.r.t. $N(f)$. For LFP, starting with $z_0 = -\infty \in {\mathbb{Z}_\infty}^k$, an increasing sequence $(z_\alpha)_\alpha$ of ${\mathbb{Z}_\infty}^k$, indexed by ordinal numbers, is defined by $z_{\alpha+1} = f(x, z_\alpha)$ for $\alpha = 0, 1, \ldots$. It is clear that we have $z_{\alpha+1} = z_\alpha$ for some $\alpha \leq \omega^k$ and $\mathrm{LFP}(f)(x) = z_\alpha$. However, to complete the computation with finite repetitions, we need some acceleration technique, which is the main topic of this paper.

Because LFP and GFP are dual, we mainly concentrate on LFP. Assume $f \in \mathscr{F}$, $f : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^m$, $c \in {\mathbb{Z}_\infty}^m$, and $f(c) \geq c$. There exists the least y such that $y \geq c$ and $f(x,y) = y$. This y is denoted by $\mathrm{LFP}_c(f)$. For $\mathrm{LFP}(f)$, it is sufficient to compute $\mathrm{LFP}_c(f)$, because $\mathrm{LFP}(f) = \mathrm{LFP}_{-\infty}(f)$. We observe that if the operator max does not appear in the definition sequence of f, then $\mathrm{LFP}_c(f)$ can be computed relatively easily. To formalize the observation, we introduce a concept called "steplessness." Let $T \subseteq I_n$. A function $f : {\mathbb{Z}_\infty}^n \to \mathbb{Z}_\infty$ is *upward stepless* w.r.t. T if f is monotone w.r.t. T, and for all $k \in T$ and $x \in {\mathbb{Z}_\infty}^n$, $f(x) = f(x + z^n_{k,1})$ implies $f(x) = f(x + z^n_{k,c})$ for all $c \in \mathbb{N}_\infty$. Word "upward" and set T are omitted if no confusion occurs. A function $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ is *stepless* if $\pi_j f$ is stepless for all $j \in I_m$. Functions γ_c, π_T, $\min(x,y)$, and $x +_\downarrow y$ are stepless, but $\max(x,y)$ and $x +_\uparrow y$ are not. Stepless functions are closed under compositions: more precisely, if $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^k$ is stepless w.r.t. T, $g : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^n$, $\pi_i g$ is stepless w.r.t. S for all $i \in T$, and $\pi_i g = \pi_i$ for all $i \in I_n \setminus T$, then $f \circ g$ is stepless w.r.t. S.

The least fixed-point of a stepless function is computed using the following lemma, which can be proved in a similar manner as in the corresponding lemma in [5],

Lemma 1. *Assume $f : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^m$ is stepless, $c \in {\mathbb{Z}_\infty}^m$, and $f(c) \geq c$. Let $\bar{c} = \mathrm{LFP}_c(f)$, $T = \{i \in I_m \mid \pi_i c < \pi_i \bar{c}\}$, and $S = I_m \setminus T$. Thus, with respect to the partition (T,S), we have $c = (d,e)$, $\bar{c} = (\bar{d},e)$, and $d \ll \bar{d}$. Then, the following hold.*

(1) $T = \{i \in I_m \mid \pi_i c < \pi_i f^{(m)}(c)\}$.

(2) There is $i \in I_m$ such that $\pi_i \mathrm{LFP}_c(f) = \pi_i f(\infty, e)$

(3) $\mathrm{LFP}_c(f) = f^{(|T|)}(\infty, e)$.

For GFP, we define $\mathrm{GFP}_c(f)(x)$ to be the greatest y such that $y \leq c$ and $f(x,y) = y$, if $f(x,c) \leq c$. Function f is *downward stepless* w.r.t. T if for all $k \in T$, $f(x) = f(x - z^n_{k,1})$ implies $f(x) = f(x - z^n_{k,c})$ for all $c \in \mathbb{N}_\infty$. Then, the counterpart of Lemma 1 holds: if f is downward stepless, $f(c) \leq c$, $c = (d,e)$ and $\bar{c} = (\bar{d},e) = \mathrm{GFP}_d(f)$ w.r.t. a partition (T,S), and $\bar{d} \ll d$, then we have (1) $T = \{i \in I_m \mid \pi_i d > \pi_i f^{(m)}(d)\}$, (2) there is $i \in I_m$ such that $\pi_i \bar{d} = \pi_i f(\infty, e)$, and (3) $\bar{d} = f^{|T|}(\infty, e)$.

Although not all functions in $\mathscr{F}$ are stepless, we can approximate them with stepless functions. Assume $f \in \mathscr{F}$, $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$, and $c \in {\mathbb{Z}_\infty}^n$. We define the *under approximation* $\mathrm{ua}(f,c) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ and the *over approximation* $\mathrm{oa}(f,c) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ as shown in Figure 1. The intuition is as follows: we wish to define $\mathrm{ua}(f,c)$ to be an upward stepless function. Because pair, minimum, $+_\downarrow$ preserves steplessness (because it is closed under compositions), these operations can be handled naturally. LFP and GFP are also all right, because they are expressed as compositions if the operand is stepless, by Lemma 1 (3). Operation $+_\uparrow$ does not preserve steplessness, but it is almost the same as operation $+_\downarrow$, and the exceptional case can be covered by a constant function. Finally, for max, we simply choose one of the operands, by referring their values at c.

The following lemma can be proved without difficulty.

$$\begin{array}{l}
\mathrm{ua}(\gamma_e,c)=\gamma_e\\
\mathrm{ua}(\pi_T,c)=\pi_T\\
\mathrm{ua}(-f,c)=-\mathrm{oa}(f,c)\\
\mathrm{ua}((f,g),c)=(\mathrm{ua}(f,c),\mathrm{ua}(g,c))\\
\mathrm{ua}(\min(f,g),c)=\min(\mathrm{ua}(f,c),\mathrm{ua}(g,c))\\
\mathrm{ua}(\max(f,g),c)=\begin{cases}\mathrm{ua}(f,c) & \text{if } f(c)\geq g(c)\\ \mathrm{ua}(g,c) & \text{otherwise}\end{cases}\\
\mathrm{ua}(f+_{\downarrow}g,c)=\mathrm{ua}(f,c)+_{\downarrow}\mathrm{ua}(g,c)\\
\mathrm{ua}(f+_{\uparrow}g,c)=\\
\begin{cases}\gamma_{-\infty} & \text{if } \{f(c),g(c)\}=\{\infty,-\infty\}\\ \mathrm{ua}(f,c)+_{\downarrow}\mathrm{ua}(g,c) & \text{otherwise}\end{cases}\\
\mathrm{ua}(\mathrm{LFP}(f),c)=\mathrm{LFP}_c(\mathrm{ua}(f,(c,\mathrm{LFP}(f)(c))))\\
\mathrm{ua}(\mathrm{GFP}(f),c)=\mathrm{GFP}_c(\mathrm{ua}(f,(c,\mathrm{GFP}(f)(c))))
\end{array}$$

$$\begin{array}{l}
\mathrm{oa}(\gamma_e,c)=\gamma_e\\
\mathrm{oa}(\pi_T,c)=\pi_T\\
\mathrm{oa}(-f,c)=-\mathrm{ua}(f,c)\\
\mathrm{oa}((f,g),c)=(\mathrm{oa}(f,c),\mathrm{oa}(g,c))\\
\mathrm{oa}(\min(f,g),c)=\begin{cases}\mathrm{oa}(f,c) & \text{if } f(c)\leq g(c)\\ \mathrm{oa}(g,c) & \text{otherwise}\end{cases}\\
\mathrm{oa}(\max(f,g),c)=\max(\mathrm{oa}(f,c),\mathrm{oa}(g,c))\\
\mathrm{oa}(f+_{\downarrow}g,c)=\\
\begin{cases}\gamma_{\infty} & \text{if } \{f(c),g(c)\}=\{\infty,-\infty\}\\ \mathrm{oa}(f,c)+_{\uparrow}\mathrm{oa}(g,c) & \text{otherwise}\end{cases}\\
\mathrm{oa}(f+_{\uparrow}g,c)=\mathrm{oa}(f,c)+_{\uparrow}\mathrm{ua}(g,c)\\
\mathrm{oa}(\mathrm{LFP}(f),c)=\mathrm{LFP}_c(\mathrm{oa}(f,(c,\mathrm{LFP}(f)(c))))\\
\mathrm{oa}(\mathrm{GFP}(f),c)=\mathrm{GFP}_c(\mathrm{oa}(f,(c,\mathrm{GFP}(f)(c))))
\end{array}$$

Figure 1: Under/Over Approximation

Lemma 2. *Assume $f\in\mathscr{F}$, $f:\mathbb{Z}_{\infty}{}^{n}\to\mathbb{Z}_{\infty}{}^{m}$, and $c\in\mathbb{Z}_{\infty}{}^{n}$.*

(1) $\mathrm{ua}(f,c)$ is upward stepless w.r.t. $P(f)$, and $\mathrm{oa}(f,c)$ is downward stepless w.r.t. $P(f)$.

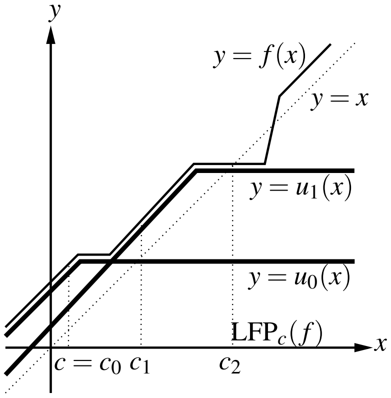
(2) $\mathrm{ua}(f,c)(c)=\mathrm{oa}(f,c)(c)=f(c)$.

(3) $\mathrm{ua}(f,c)(x)\leq f(x)\leq\mathrm{oa}(f,c)(x)$ for all $x\in\mathbb{Z}_{\infty}{}^{n}$.

(4) If $f(c)\geq c$, then $\mathrm{LFP}_c(\mathrm{ua}(f,c))\leq\mathrm{LFP}_c(f)$. If $f(c)\leq c$, then $\mathrm{GFP}_c(\mathrm{oa}(f,c))\geq\mathrm{GFP}_c(f)$.

(5) $\{\mathrm{ua}(f,d)\mid d\in\mathbb{Z}_{\infty}{}^{n}\}$ and $\{\mathrm{oa}(f,d)\mid d\in\mathbb{Z}_{\infty}{}^{n}\}$ are finite sets.

3 Procedure



Based on the preparation in the previous section, we now describe the procedure to compute $\mathrm{LFP}_c(f)$ for $f\in\mathscr{F}$ and $c\in\mathbb{Z}_{\infty}{}^{m}$ such that $f:\mathbb{Z}_{\infty}{}^{m}\to\mathbb{Z}_{\infty}{}^{m}$ and $f(c)\geq c$. As a starting point, we consider the following naive procedure: starting with $c_0=c$, we compute $c_{n+1}=\mathrm{LFP}_{c_n}(\mathrm{ua}(f,c_n))$ until $c_n=c_{n+1}$. Because $\mathrm{ua}(f,c_n)$ is stepless, we can compute c_{n+1} using Lemma 1. Then, $\mathrm{LFP}_c(f)=c_n$.

The right figure illustrates the intuition behind the procedure. Note that $\bar{c}=\mathrm{LFP}_c(f)$ is the left-most intersection (on the right side of c) of the graph of $y=f(x)$ with that of $y=x$. Because of Lemma 2 (4), $c_1=\mathrm{LFP}_{c_0}(u_0)$ is smaller than or equal to $\bar{c}$, and c_1 is greater than or equal to c_0 by its definition. Thus, we have $c_0\leq c_1\leq\cdots$. The number of repetitions seems to be finite, because of Lemma 2 (5) and the fact that $\mathrm{ua}(f,c_n)$ becomes "constant" on the right side of c_{n+1}.

Unfortunately, the intuition is not correct. The procedure may not terminate if f has two or more parameters, as will be shown in Example 5. To resolve the problem, the under approximation should be taken component-wise, and if f does not move a component of c_n, we keep the previous approximation for that component.

The modified procedure is shown in Figure 2. For $f\in\mathscr{F}$, $\mathrm{LFP}_c(f)$ is computed in the left column. Because u_n is stepless, $\mathrm{LFP}_{c_n}(u_n)$ appearing in the left column is computed in the right column.

Procedure for $f \in \mathscr{F}$.
Compute $c_n \in \mathbb{Z}_\infty{}^m$ and define stepless function u_n until $c_{n+1} = c_n$:

$$c_0 = c, \quad u_0 = \mathrm{ua}(f, c_0).$$

$$c_{n+1} = \mathrm{LFP}_{c_n}(u_n).$$

$$\pi_i u_{n+1} = \begin{cases} \pi_i u_n & \text{if } \pi_i f(c_{n+1}) = \pi_i c_{n+1} \\ \mathrm{ua}(\pi_i f, c_{n+1}) & \text{otherwise} \end{cases}$$

Then, $\mathrm{LFP}_c(f) = c_n$.

Procedure for stepless f.
Compute $d_n \in \mathbb{Z}_\infty{}^m$ until $T_{n+1} = T_n$, where $T_n = \{i \in I_n \mid \pi_i d_n > \pi_i c\}$:

$$d_0 = c, \quad d_{n+1} = f(d_n).$$

Let $T = T_n$, $S = I_m \setminus T$, and $s = \pi_S c$. Compute $e_k \in \mathbb{Z}_\infty{}^m$ until $e_{k+1} = e_k$:

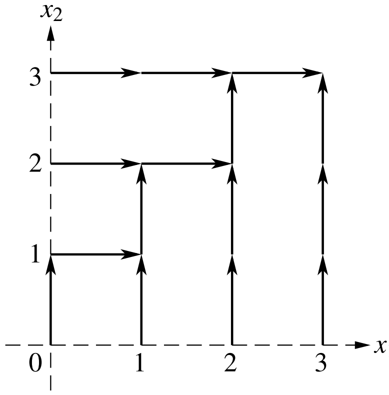
$$e_0 = (\infty, s), \quad e_{k+1} = f(e_k)$$

Then, $\mathrm{LFP}_c(f) = e_k$.

Figure 2: Procedure to Compute $\mathrm{LFP}_c(f)$

Example 3. Let $f : \mathbb{Z}_\infty{}^2 \to \mathbb{Z}_\infty{}^2$ be defined by $f(x_1, x_2) = (x_1, \min(x_1 +_\downarrow x_2, 10))$. We compute $\mathrm{LFP}_{(1,0)}(f)$. Because f is stepless, the right column of Figure 2 is used. The computation of the first half is as follows: $d_0 = (1,0)$, $T_0 = \varnothing$, $d_1 = (1,1)$, $T_1 = \{2\}$, $d_2 = (1,2)$. $T_2 = \{2\}$. Here, we have $T_1 = T_2 = \{2\}$. With this result, we start the second half: $e_0 = (1,\infty)$, $e_1 = (1,10)$, $e_2 = (1,10)$. Thus, we get the result: $\mathrm{LFP}_{(1,0)}(f) = (1,10)$.

Example 4. Assume that $n \in \mathbb{N}$ and $f_n : \mathbb{Z}_\infty \to \mathbb{Z}_\infty$ is defined by $f_n(x) = \max(n+1+\min(x-n,0),0)$. We compute $\mathrm{LFP}(f_n)$. First, $c_0 = -\infty$ and $u_0 = \gamma_0$, because $n+1+\min(-\infty-n,0) < 0$. Therefore, $c_1 = \mathrm{LFP}_{-\infty}(\gamma_0) = 0$. Because $f_n(c_1) = 1 \neq c_1$, we take $u_1(x) = \mathrm{ua}(f_n,0)(x) = n+1+\min(x-n,0)$. Using the right column, we get $c_2 = \mathrm{LFP}_0(u_1) = n+1$. Repeating this step, we find $c_3 = n+1$, and conclude $\mathrm{LFP}(f_n) = n+1$.



Example 5. Let function $f : \mathbb{Z}_\infty{}^2 \to \mathbb{Z}_\infty{}^2$ be defined by $f(x_1, x_2) = (\min(x_1+1, \max(x_1,x_2)), \min(\max(x_2, x_1+1), x_2+1))$. A small calculation could show that $f(x_1,x_2) = (x_1, x_2+1)$ if $x_1 \geq x_2$, and $f(x_1,x_2) = (x_1+1, x_2)$ if $x_1 < x_2$, although it is not a part of the procedure. In the right figure, f is depicted by arrows connecting (x_1,x_2) and $f(x_1,x_2)$. It is clear that $\mathrm{LFP}_{(0,0)}(f) = (\infty,\infty)$.

If we apply the naive procedure shown in the beginning of this section, the computation does not terminate. We start with $c_0 = (0,0)$ and $u_0(x_1,x_2) = (x_1, \min(x_1+1, x_2+1))$. Then, $c_1 = \mathrm{LFP}_{c_0}(u_0) = (0,1)$, and $u_1(x_1,x_2) = (\min(x_1+1,x_2), x_2)$. In the next steps, we have $c_2 = (1,1)$, $u_2 = u_0$, $c_3 = (1,2)$, and $u_3 = u_1$. Thus, the computation continues for ever, calculating $c_{2n} = (n,n)$, $c_{2n+1} = (n, n+1)$, $u_{2n} = u_0$, and $u_{2n+1} = u_1$.

On the other hand, the modified procedure does terminate. The computation goes in the same way up to c_1. When we decide u_1, two values $c_1 = (0,1)$ and $f(c_1) = (1,1)$ are compared. Because their x_2-components are identical, we reuse the x_2-component of u_0 for that of u_1. The x_1-component is calculated as usual. Thus, we have $u_1(x_1,x_2) = (\min(x_1+1,x_2), \min(x_1+1,x_2+1))$, and because $\mathrm{LFP}_{c_1}(u_1) = (\infty,\infty)$, we reach the result in finite repetitions.

The partial correctness of the procedure is almost clear from the previous lemmas. When f is stepless, by Lemma 1 (1), $(T, I_m \setminus T)$ gives the required partition, where $T = T_n = T_{n+1}$. Then, Lemma 1 (3) guarantees that $\mathrm{LFP}_c(f) = e_k$. For $f \in \mathscr{F}$, Lemma 1 shows that the right column computes the fixed-point for stepless functions. By Lemma 2 (4), $(c_n)_n$ is an increasing sequence that does not exceed $\mathrm{LFP}_c(f)$. Therefore, if we reach n such that $c_{n+1} = c_n$, then $\mathrm{LFP}_c(f) = c_n$.

To prove the termination, i.e., there exists n such that $c_{n+1} = c_n$, we use the following technical lemma.

Lemma 6. *Assume (T,S) is a partition of I_m, $u = (v,w) : \mathbb{Z}_\infty{}^m \to \mathbb{Z}_\infty{}^m$ is a stepless function, $(a_2,b_2) = \mathrm{LFP}_{(a_1,b_1)}u$, $v(a_2,b_3) = a_2$, $a_1 \ll a_2$, and $b_2 \ll b_3$. Then, there exists $i \in T$ such that $\pi_i a_2 = \pi_i v(\infty,\infty)$.*

Proof. Let $t = |T|$, $\bar{v} : \mathbb{Z}_\infty{}^t \to \mathbb{Z}_\infty{}^t$ be defined by $\bar{v}(a) = v(a,\infty)$, and $\bar{a}_2 = \mathrm{LFP}_{a_1}\bar{v}$. We have $a_2 \leq \bar{a}_2$: this is because while a_2 is the T-part of the supremum of the sequence $(x_\alpha)_\alpha$ defined by $x_0 = (a_1,b_1)$ and $x_{\alpha+1} = v(x_\alpha)$, $\bar{a}_2$ is the supremum of the sequence $(y_\alpha)_\alpha$ defined by $y_0 = a_1$ and $y_{\alpha+1} = v(y_\alpha,\infty)$. We can show that $y_\alpha \geq \pi_T x_\alpha$ by induction on α. On the other hand, because $v(a_2,b_2) = v(a_2,b_3)$, $b_2 \ll b_3$, and v is stepless, we have $v(a_2,\infty) = a_2$, i.e., a_2 is a fixed point of $\bar{v}$. Therefore, $a_2 = \bar{a}_2$. Because $\bar{v}$ is stepless and $a_1 \ll a_2$, by Lemma 1 (2), there exists $i \in T$ such that $\pi_i a_2 = \pi_i \bar{v}(\infty) = \pi_i v(\infty,\infty)$. □

We sketch a termination proof. Assume on the contrary that the sequence does not converge: $c_0 < c_1 < \cdots < c_n < c_{n+1} < \cdots$. Let $U = \{i \in I_m \mid \pi_i c_n < \pi_i c_{n+1}$ for infinitely many $n\}$ and $V = I_m \setminus U$. There exists $N \in \mathbb{N}$ such that for all $n \geq N$, $\pi_V c_n = \pi_V c_N$. Let $s = \pi_V c_N$. We write $c_n = (e_n, s)$.

We claim that for any $n' \geq N$, there exists $n \geq n'$ and $i \in U$ such that $e_{n'} \ll e_n$ and $\pi_i u_n(\infty,s) = \pi_i e_n$. Let k be the least k such that $e_{n'} \ll e_k$, and m be the largest m such that $e_m \ll e_k$. Let $T = \{i \in U \mid \pi_i e_{m+1} = \pi_i e_k\}$ and $S = U \setminus T$. By applying Lemma 6 with $u := u_m$ (with V-part fixed to s), $(a_1,b_1) := e_m$, $(a_2,b_2) := e_{m+1}$, and $b_3 := \pi_S e_k$, we confirm the claim by taking $n = m+1$.

By Lemma 2 (5), there exists $i \in T$ and $n,k \in \mathbb{N}$ such that $e_n \ll e_k$, $u_n = u_k$, $\pi_i u_n(\infty,s) = \pi_i e_n$, and $\pi_i u_k(\infty,s) = \pi_i e_k$, which is impossible.

4 Function Examples

In this section, we show that several functions defined on graphs are regarded as elements of $\mathscr{F}$, thus they can be computed by our algorithm. We do not claim that our procedure is more suitable to calculate values of these particular functions than known algorithms. Instead, these examples illustrate various quantitative properties are expressed as the fixed-point of a function in $\mathscr{F}$.

4.1 Shortest Path

Assume that $G = \{s_1, s_2, \ldots, s_n\}$ is a finite set of nodes and the lengths $d(i,j) \in \mathbb{N}_\infty$ of the connection between two adjacent nodes s_i and s_j are given. If s_i and s_j are not directly connected, $d(i,j) = \infty$. We fix $s = s_{i_0} \in G$, and define f by $\pi_i f(x) = \min(\{d(i,i_0)\} \cup \{d(i,j) + \pi_j x \mid j \neq i_0\})$. Then, the length of the shortest path between s_i to s is $\pi_i \mathrm{GFP}_x(f)$ (or $\pi_i \mathrm{LFP}_x(f)$: they coincide in this case). The intuitive meaning of this definition is that the shortest path from s_i is either the direct connection to s_{i_0} or a path to some adjacent state s_j connected with the shortest path between s_j and s_{i_0}, whichever is the shortest.

If (G,E) is a graph and $d(i,j)$ is defined by $d(i,j) = 1$ if $(s_i,s_j) \in E$, $d(i,j) = \infty$ otherwise, then the same function computes the shortest hop count from s_i to s_{i_0}.

4.2 Proof-Number Search

Let us consider a finite tree with labels on nodes. The label is either ‘true’, ‘false’, or ‘unknown’ on a leaf node, and either ‘MAX’ or ‘MIN’ on an internal node. The proof number proof(n) of node n is defined as follows [1]: the value of proof(n) is 0 if the label is ‘true’, ∞ if ‘false’, and 1 if ‘unknown’. For an internal node, proof$(n) = \min\{\text{proof}(n') \mid n' \in \text{child}(n)\}$ if the label is ‘MAX’, and proof$(n) = \sum\{\text{proof}(n') \mid n' \in \text{child}(n)\}$ if the label is ‘MIN’.

The proof number can be computed as a value of function in $\mathscr{F}$: let $\{n_i \mid i = 1,\ldots,N\}$ be an enumeration of the nodes of a tree. For each i, we define $f_i : \mathbb{Z}_\infty{}^N \to \mathbb{Z}_\infty$ to reflect the definition of the proof number. For example, if s_i is a leaf node with label ‘false,’ $f_i(x) = \infty$ for any x, and if s_i is an internal node with label 'MIN,' $f_i(x) = \sum(x_j \mid s_j \in \mathrm{child}(s_i))$. Let $f : \mathbb{Z}_\infty{}^N \to \mathbb{Z}_\infty{}^N$ be such that $\pi_i f = f_i$ for all i. Then, it is obvious that $\mathrm{proof}(n_i) = \pi_i \mathrm{GFP}(f)$.

4.3 Data Flow Analysis

Lacey et al. used CTL to specify conditions on a control flow graph by regarding the graph as a Kripke structure for the purpose of program transformation [7]. We extended the approach by introducing a non-standard semantics of the modal μ-calculus [4]. For example, let us introduce a propositional symbol **accessx** that holds at a node in a control flow graph if the program variable **x** is accessed at the node. The minimum number of accesses to the variable **x** on an execution path starting from a node can then be expressed by the following formula under the non-standard semantics:

$$\nu X(((\mathbf{accessx} \wedge 1) \vee (\neg \mathbf{accessx})) \wedge (\mathbf{halt} \vee \Diamond X)).$$

Here, **halt** is an abbreviation for $\Box\bot$, which means that no outgoing transition exists.

Now, for given Kripke structure $\mathscr{K} = (S,R,L)$, we enumerate $S = \{s_1,\ldots,s_n\}$ and denote the set $\{s' \mid (s,s') \in R\}$ by sR. For $i \in I_n$, we define $a_i, h_i \in \mathbb{Z}_\infty$ by $a_i = 1$ if **x** is accessed at s_i, $a_i = 0$ otherwise, and $h_i = 0$ if there is no outgoing node at s_i, $h_i = \infty$ otherwise. Then, the above formula corresponds to function $\mathrm{LFP}(f)$ in $\mathscr{F}$, where $f : \mathbb{Z}_\infty{}^n \to \mathbb{Z}_\infty{}^n$ is defined as follows,

$$\pi_i f(x) = a_i + \min(h_i, \min\{\pi_j x \mid s_j \in s_i R\}).$$

5 Conclusion

5.1 Remark on Efficiency

Compared to the algorithm proposed in [4], the procedure in this paper not only covers wider range of functions, but also is more efficient in some cases. For example, The old algorithm requires $O(n)$ time to compute the least fixed-point of f_n in Example 4 (more precisely, their corresponding functions defined on $\mathbb{N}_\infty$), but the algorithm in Section 3 requires constant time.

Unfortunately, we have not yet obtained the time complexity of the algorithm. Even if we restrict ourselves to fixed-point free functions, the current termination proof shown in Section 3 does not provide the number of required repetitions. To evaluate the complexity of the algorithm remains as future work.

Acknowledgments

The authors would like to thank the reviewers for their careful reading and helpful comments.

This research has been partially supported by Grant-in-Aid for Scientific Research by Ministry of Education, Culture, Science and Technology, Scientific Research(C) 21500006, "Decision procedures of modal logics and their application to software verification."

References

[1] L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artif. Intell.*, 66(1):91–124, 1994.

[2] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons, 1992.

[3] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[4] D. Ikarashi, Y. Tanabe, K. Nishizawa, and M. Hagiya. Modal μ-calculus on min-plus algebra $\mathbb{N}_\infty$. In *Proc. of 10th Wksh. on Programming and Programming Languages, PPL 2008 (March 2008)*, 2008. Available at `http://www.nue.riec.tohoku.ac.jp/ppl2008/program.html`.

[5] D. Ikarashi, Y. Tanabe, K. Nishizawa, and M. Hagiya. Modal μ-calculus on min-plus algebra $\mathbb{N}_\infty$. Revised version of [4], submitted, 2009. Available at `http://cent.xii.jp/tanabe.yoshinori/09/05/minplusPC.pdf`.

[6] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27(3):333–354, 1983.

[7] D. Lacey, N. D. Jones, E. Van Wyk, and C. C. Frederiksen. Compiler optimization correctness by temporal logic. *Higher-Order and Symb. Comput.*, 17(3):173–206, 2004.

[8] I. Simon. Limited subsets of a free monoid. In *Proc. of 19th Ann. Symp. on Foundations of Computer Science, FOCS '78 (Ann Arbor, MI, Oct. 1978)*, pp. 143–150. IEEE CS Press, 1978.

A Non-Uniform Finitary Relational Semantics of System T

Lionel Vaux[*]
Laboratoire de Mathématiques de l'Université de Savoie
UFR SFA, Campus Scientifique, F-73376 Le Bourget-du-Lac Cedex, France
lionel.vaux@univ-savoie.fr

Abstract

We study iteration and recursion operators in the denotational semantics of typed λ-calculi derived from the multiset relational model of linear logic. Although these operators are defined as fixpoints of typed functionals, we prove them finitary in the sense of Ehrhard's finiteness spaces.

1 Introduction

Finiteness spaces were introduced by Ehrhard [2], refining the purely relational model of linear logic. A finiteness space is a set equipped with a finiteness structure, i.e. a particular set of subsets which are said to be finitary; and the model is such that the relational denotation of a proof in linear logic is always a finitary subset of its conclusion. By the usual co-Kleisli construction, this also provides a model of the simply typed λ-calculus: the cartesian closed category $\underline{\mathrm{Fin}}$. The main property of finiteness spaces is that the intersection of two finitary subsets of dual types is always finite. This feature allows to reformulate Girard's quantitative semantics [7] in a standard algebraic setting, where morphisms interpreting typed λ-terms are analytic functions between the topological vector spaces generated by vectors with finitary supports. This provided the semantical foundations of Ehrhard-Regnier's differential λ-calculus [4] and motivated the general study of a differential extension of linear logic (e.g., [5, 6, 3, 13, 14, 11, 10]).

It is worth noticing that finiteness spaces can accomodate typed λ-calculi only. In particular, the relational semantics of fixpoint combinators is never finitary. The whole point of the finiteness construction is actually to reject infinite computations, ensuring the intermediate sets involved in the relational interpretation of a cut are all finite. Despite this restrictive design, Ehrhard proved that a limited form of recursion was available, by defining a finitary tail-recursive iteration operator.

The main result of the present paper is that finiteness spaces can actually accomodate the standard notion of primitive recursion in λ-calculus, Gödel's system T: we prove $\underline{\mathrm{Fin}}$ admits a weak natural number object in the sense of [12, 9], and we more generally exhibit a finitary recursion operator for this interpretation of the type of natural numbers. This achievement is twofold:

- Before considering finiteness, we must define a recursion operator in the cartesian closed category deduced from the relational model of linear logic. For that purpose, we cannot follow Ehrhard and use the flat interpretation of the type Nat of natural numbers. Indeed, if t, u and v are terms of types respectively Nat, $\mathsf{Nat} \Rightarrow X \Rightarrow X$ and X, the recursion step $\mathsf{R}\,(\mathsf{S}\,t)\,u\,v \rightsquigarrow u\,t\,(\mathsf{R}\,t\,u\,v)$ puts t in argument position. In case u is a constant function, t is not used in the reduced form. The recursor R must however discriminate between $\mathsf{S}\,t$ and O, hence the successor S cannot be linear: it must produce information independently from its input. Though it might be obscure for the reader not familiar with the relational or coherence semantics, this argument will be made formal in the paper. This was already noted by Girard in coherence spaces [8]: we adopt the solution he proposed, and interpret terms of type Nat by so-called *lazy* natural numbers. An notable outcome is that our interpretation provides a semantic evidence of the well-known gap in expressive power between the iterator and recursor variants of system T.

[*] This work has been partially funded by the French ANR projet blanc "Curry Howard pour la Concurrence" CHOCO ANR-07-BLAN-0324.

- The second aspect of our work is to establish that this relational semantics is finitary. This is far from immediate because the recursion operator is defined as the fixpoint of finitary approximants: since fixpoints themselves are not finitary relations, it is necessary to obtain stronger properties of these approximants to conclude.

Structure of the paper. In section 2, we briefly describe two cartesian closed categories: the category $\underline{\mathrm{Rel}}$ of sets and relations from multisets to points, and the category $\underline{\mathrm{Fin}}$ of finiteness spaces and finitary relations from multisets to points. In section 3, we give an explicit presentation of the relational semantics of typed λ-calculi in $\underline{\mathrm{Rel}}$ and $\underline{\mathrm{Fin}}$, which we extend to system T in section 4. In section 5, we establish a uniformity property of iteration-definable morphisms, which does not hold for recursion in general.

2 Sets, Relations and Finiteness Spaces

If A is a set, denote by $\mathfrak{P}(A)$ the powerset of A, by $\mathfrak{P}_{\mathrm{f}}(A)$ the set of all finite subsets of A and by $A^{!}$ the set of all finite multisets of A. If $(\alpha_1,\dots,\alpha_n)\in A^n$, we write $\overline{\alpha}=[\alpha_1,\dots,\alpha_n]$ for the corresponding multiset, and denote multiset union additively. Let $f\subseteq A\times B$ be a relation from A to B, we write $f^{\perp}=\{(\beta,\alpha);\ (\alpha,\beta)\in f\}$. For all $a\subseteq A$, we set $f\cdot a=\{\beta\in B;\ \exists\alpha\in a,\ (\alpha,\beta)\in f\}$. We write $\underline{\mathrm{Rel}}$ for the coKleisli category of the comonad $(-)^{!}$ in the relational model of linear logic (see e.g. [1]): objects are sets and $\underline{\mathrm{Rel}}(A,B)=\mathfrak{P}\left(A^{!}\times B\right)$; the identity on A is $id_A=\{([\alpha],\alpha);\ \alpha\in A\}$; if $f\in\underline{\mathrm{Rel}}(A,B)$ and $g\in\underline{\mathrm{Rel}}(B,C)$ then $g\circ f=\left\{(\sum_{i=1}^{n}\overline{\alpha}_i,\gamma);\ \exists\overline{\beta}=[\beta_1,\dots,\beta_n]\in B^{!},\ (\overline{\beta},\gamma)\in g\wedge\forall i\ (\overline{\alpha}_i,\beta_i)\in f\right\}$.

The category $\underline{\mathrm{Rel}}$ is cartesian closed. The cartesian product is given by the disjoint union of sets $A\uplus B=(\{1\}\times A)\cup(\{2\}\times B)$, with terminal object the empty set $\emptyset$. Projections are $\{([(1,\alpha)],\alpha);\ \alpha\in A\}\in\underline{\mathrm{Rel}}(A\uplus B,A)$ and $\{([(2,\beta)],\beta);\ \beta\in B\}\in\underline{\mathrm{Rel}}(A\uplus B,B)$. If $f\in\underline{\mathrm{Rel}}(C,A)$ and $g\in\underline{\mathrm{Rel}}(C,B)$, pairing is given by: $\langle f,g\rangle=\{(\overline{\gamma},(1,\alpha));\ (\overline{\gamma},\alpha)\in f\}\cup\{(\overline{\gamma},(2,\beta));\ (\overline{\gamma},\beta)\in g\}\in\underline{\mathrm{Rel}}(C,A\uplus B)$. The unique morphism from A to $\emptyset$ is $\emptyset$. The adjunction for closedness is $\underline{\mathrm{Rel}}(A\uplus B,C)\cong\underline{\mathrm{Rel}}(A,B^{!}\times C)$ which boils down to the bijection $(A\uplus B)^{!}\cong A^{!}\times B^{!}$.

We recall the few notions we shall use on finiteness spaces. For a detailled presentation, the obvious reference is [2]. Let $\mathfrak{F}\subseteq\mathfrak{P}(A)$ be any set of subsets of A. We define the pre-dual of $\mathfrak{F}$ in A as $\mathfrak{F}^{\perp}=\{a'\subseteq A;\ \forall a\in\mathfrak{F},\ a\cap a'\in\mathfrak{P}_{\mathrm{f}}(A)\}$. By a standard argument, we have the following immediate properties: $\mathfrak{P}_{\mathrm{f}}(A)\subseteq\mathfrak{F}^{\perp}$; $\mathfrak{F}\subseteq\mathfrak{F}^{\perp\perp}$; if $\mathfrak{G}\subseteq\mathfrak{F}$, $\mathfrak{F}^{\perp}\subseteq\mathfrak{G}^{\perp}$. By the last two, we get $\mathfrak{F}^{\perp}=\mathfrak{F}^{\perp\perp\perp}$. A *finiteness structure* on A is a set $\mathfrak{F}$ of subsets of A such that $\mathfrak{F}^{\perp\perp}=\mathfrak{F}$. Then a *finiteness space* is a dependant pair $\mathscr{A}=(|\mathscr{A}|,\mathfrak{F}(\mathscr{A}))$ where $|\mathscr{A}|$ is the underlying set, called the *web* of $\mathscr{A}$, and $\mathfrak{F}(\mathscr{A})$ is a finiteness structure on $|\mathscr{A}|$. We write $\mathscr{A}^{\perp}$ for the dual finiteness space: $\left|\mathscr{A}^{\perp}\right|=|\mathscr{A}|$ and $\mathfrak{F}\left(\mathscr{A}^{\perp}\right)=\mathfrak{F}(\mathscr{A})^{\perp}$. The elements of $\mathfrak{F}(\mathscr{A})$ are called the *finitary subsets* of $\mathscr{A}$.

For all set A, $(A,\mathfrak{P}_{\mathrm{f}}(A))$ is a finiteness space and $(A,\mathfrak{P}_{\mathrm{f}}(A))^{\perp}=(A,\mathfrak{P}(A))$. In particular, each finite set A is the web of exactly one finiteness space: $(A,\mathfrak{P}_{\mathrm{f}}(A))=(A,\mathfrak{P}(A))$. We introduce the empty finiteness space $\mathscr{T}=(\emptyset,\{\emptyset\})$ and the finiteness space of *flat natural numbers* $\mathscr{N}=(\mathbf{N},\mathfrak{P}_{\mathrm{f}}(\mathbf{N}))$. If $\mathscr{A}$ and $\mathscr{B}$ are finiteness spaces, we define $\mathscr{A}\,\&\,\mathscr{B}$ and $\mathscr{A}\Rightarrow\mathscr{B}$ as follows. Let $|\mathscr{A}\,\&\,\mathscr{B}|=|\mathscr{A}|\uplus|\mathscr{B}|$ and $\mathfrak{F}(\mathscr{A}\,\&\,\mathscr{B})=\{a\uplus b;\ a\in\mathfrak{F}(\mathscr{A})\wedge b\in\mathfrak{F}(\mathscr{B})\}$. Let $|\mathscr{A}\Rightarrow\mathscr{B}|=|\mathscr{A}|^{!}\times|\mathscr{B}|$ and set $f\in\mathfrak{F}(\mathscr{A}\Rightarrow\mathscr{B})$ iff: $\forall a\in\mathfrak{F}(\mathscr{A})$, $f\cdot a^{!}\in\mathfrak{F}(\mathscr{B})$, and $\forall\beta\in|\mathscr{B}|$, $(f^{\perp}\cdot\{\beta\})\cap a^{!}$ is finite. It is easily seen that $\mathscr{A}\,\&\,\mathscr{B}$ is a finiteness space, but the same result for $\mathscr{A}\Rightarrow\mathscr{B}$ is quite technical and the only known proof uses the axiom of choice [2]. We call *finitary relations* the elements of $\mathfrak{F}(\mathscr{A}\Rightarrow\mathscr{B})$.

Notice that $\mathfrak{F}(\mathscr{A}\Rightarrow\mathscr{B})\subseteq\underline{\mathrm{Rel}}(|\mathscr{A}|,|\mathscr{B}|)$. We write $\underline{\mathrm{Fin}}$ for the category of finiteness spaces with $\underline{\mathrm{Fin}}(\mathscr{A},\mathscr{B})=\mathfrak{F}(\mathscr{A}\Rightarrow\mathscr{B})$ and composition defined as in $\underline{\mathrm{Rel}}$. It is cartesian closed with terminal object $\mathscr{T}$, product $-\,\&\,-$ and exponential $-\Rightarrow-$: the definitions of those functors on morphisms, the natural transformations, and the adjunction required for cartesian closedness are exactly the same as for $\underline{\mathrm{Rel}}$.

$$\frac{}{\Gamma, x:A, \Delta \vdash x:A}\;\text{(Var)} \qquad \frac{}{\Gamma \vdash \langle\rangle : \top}\;\text{(Unit)} \qquad \frac{a \in \mathfrak{C}_A}{\Gamma \vdash a:A}\;\text{(Const)}$$

$$\frac{\Gamma, x:A \vdash s:B}{\Gamma \vdash \lambda x s : A \to B}\;\text{(Abs)} \qquad \frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t:A}{\Gamma \vdash st : B}\;\text{(App)}$$

$$\frac{\Gamma \vdash s:A \qquad \Gamma \vdash t:B}{\Gamma \vdash \langle s,t\rangle : A \times B}\;\text{(Pair)} \qquad \frac{\Gamma \vdash s : A \times B}{\Gamma \vdash \pi_l\, s : A}\;\text{(Left)} \qquad \frac{\Gamma \vdash s : A \times B}{\Gamma \vdash \pi_r\, s : B}\;\text{(Right)}$$

Figure 1: Rules of typed λ-calculi with products

$$\frac{}{\Gamma^{[]}, x^{[\alpha]} : A, \Delta^{[]} \vdash x^{\alpha} : A}\;\llbracket\text{Var}\rrbracket \qquad \frac{a \in \mathfrak{C}_A \quad \alpha \in \llbracket a \rrbracket}{\Gamma^{[]} \vdash a^{\alpha} : A}\;\llbracket\text{Const}\rrbracket$$

$$\frac{\Gamma, x^{\overline{\alpha}} : A \vdash s^{\beta} : B}{\Gamma \vdash \lambda x s^{(\overline{\alpha},\beta)} : A \to B}\;\llbracket\text{Abs}\rrbracket \qquad \frac{\Gamma_0 \vdash s^{([\alpha_1,\dots,\alpha_k],\beta)} : A \to B \qquad \Gamma_1 \vdash t^{\alpha_1} : A \quad \cdots \quad \Gamma_k \vdash t^{\alpha_k} : A}{\sum_{j=0}^{k} \Gamma_j \vdash st^{\beta} : B}\;\llbracket\text{App}\rrbracket$$

$$\frac{\Gamma \vdash s_i^{\alpha} : A_i}{\Gamma \vdash \langle s_1, s_2\rangle^{(i,\alpha)} : A_1 \times A_2}\;\llbracket\text{Pair}_i\rrbracket \qquad \frac{\Gamma \vdash s^{(1,\alpha)} : A \times B}{\Gamma \vdash \pi_l\, s^{\alpha} : A}\;\llbracket\text{Left}\rrbracket \qquad \frac{\Gamma \vdash s^{(2,\beta)} : A \times B}{\Gamma \vdash \pi_r\, s^{\beta} : B}\;\llbracket\text{Right}\rrbracket$$

Figure 2: Computing points in the relational semantics

3 The Multiset Relational Semantics of Typed λ-Calculi

Typed λ-calculi. In this section, we give an explicit description of the interpretation in Rel and Fin of the basic constructions of typed λ-calculi with products. Type and term expressions are given by:

$$A,B ::= X \mid A \to B \mid A \times B \mid \top \qquad \text{and} \qquad s,t ::= x \mid a \mid \lambda x s \mid st \mid \langle s,t\rangle \mid \pi_l\, s \mid \pi_r\, s \mid \langle\rangle$$

where X ranges over a fixed set $\mathfrak{A}$ of atomic types, x ranges over term variables and a ranges over term constants. To each variable or constant, we associate a type, and we write $\mathfrak{C}_A$ for the collection of constants of type A. A typing judgement is an expression $\Gamma \vdash s : A$ derived from the rules in Figure 1 where contexts Γ and Δ range over lists $(x_1 : A_1, \dots, x_n : A_n)$ of typed variables. The operational semantics of a typed λ-calculus is given by a contextual equivalence relation $\simeq$ on typed terms: if $s \simeq t$, then s and t have the same type, say A; we then write $\Gamma \vdash s \simeq t : A$ for any suitable Γ. In general, we will give $\simeq$ as the reflexive, symmetric and transitive closure of a contextual relation $>$ on typed terms. We define $>_0$ as the least one such that: $\pi_l\,\langle s,t\rangle >_0 s$, $\pi_r\,\langle s,t\rangle >_0 t$ and $(\lambda x s)\,t >_0 s\,[x := t]$ (with the obvious assumptions ensuring typability), and we write $\simeq_0$ for the corresponding equivalence.

Relational interpretation and finiteness property. Assume a set $\llbracket X \rrbracket$ is given for each base type X; then we interpret type constructions by $\llbracket A \to B \rrbracket = \llbracket A \rrbracket^! \times \llbracket B \rrbracket$, $\llbracket A \times B \rrbracket = \llbracket A \rrbracket \uplus \llbracket B \rrbracket$ and $\llbracket \top \rrbracket = \emptyset$. Further assume that with every constant $a \in \mathfrak{C}_A$ is associated a subset $\llbracket a \rrbracket \subseteq \llbracket A \rrbracket$. The relational semantics of a derivable typing judgement $x_1 : A_1, \dots, x_n : A_n \vdash s : A$ will be a relation $\llbracket s \rrbracket_{x_1:A_1,\dots,x_n:A_n} \subseteq \llbracket A_1 \rrbracket^! \times \cdots \times \llbracket A_n \rrbracket^! \times \llbracket A \rrbracket$. We first introduce the deductive system of Figure 2. In this system, derivable judgements are semantic annotations of typing judgements: $x_1^{\overline{\alpha}_1} : A_1, \dots, x_n^{\overline{\alpha}_n} : A_n \vdash s^{\alpha} : A$ stands for $(\overline{\alpha}_1, \dots, \overline{\alpha}_n, \alpha) \in \llbracket s \rrbracket_{x_1:A_1,\dots,x_n:A_n}$ where each $\overline{\alpha}_i \in \llbracket A_i \rrbracket^!$ and $\alpha \in \llbracket A \rrbracket$. In rules $\llbracket\text{Var}\rrbracket$ and $\llbracket\text{Const}\rrbracket$, $\Gamma^{[]}$ denotes an annotated context of the form $x_1^{[]} : A_1, \dots, x_n^{[]} : A_n$. In rule $\llbracket\text{App}\rrbracket$, the sum of annotated contexts is defined pointwise: $\left(x_1^{\overline{\alpha}_1} : A_1, \dots, x_n^{\overline{\alpha}_n} : A_n\right) + \left(x_1^{\overline{\alpha}'_1} : A_1, \dots, x_n^{\overline{\alpha}'_n} : A_n\right) = \left(x_1^{\overline{\alpha}_1 + \overline{\alpha}'_1} : A_1, \dots, x_n^{\overline{\alpha}_n + \overline{\alpha}'_n} : A_n\right)$. The semantics of a term is the set of its annotations: $\llbracket s \rrbracket_{x_1:A_1,\dots,x_n:A_n} = \left\{(\overline{\alpha}_1, \dots, \overline{\alpha}_n, \alpha);\ x_1^{\overline{\alpha}_1} : A_1, \dots, x_n^{\overline{\alpha}_n} : A_n \vdash s^{\alpha} : A\right\}$. Notice there is no rule for $\langle\rangle$ in Figure 2, hence $\llbracket \langle\rangle \rrbracket_{\Gamma} = \emptyset$ for all Γ.

Theorem 3.1 (Invariance). *If* $\Gamma \vdash s \simeq_0 t : A$ *then* $[\![s]\!]_\Gamma = [\![t]\!]_\Gamma$.

Proof. We followed the standard interpretation of typed λ-calculi in cartesian closed categories, in the particular case of Rel. A direct proof is also easy, first proving a substitution lemma: if $\Gamma_0, x : A^{[\alpha_1,\dots,\alpha_k]}, \Delta_0 \vdash s^\beta : B$, and, for all $j \in \{1,\dots,k\}$, $\Gamma_j, \Delta_j \vdash t^{\alpha_j} : A$, then $\sum_{j=0}^k \Gamma_j, \sum_{j=0}^k \Delta_j \vdash s\,[x := t]^\beta : B$. □

The relational interpretation also defines a semantics in Fin: assume a finiteness structure $\mathfrak{F}(X)$ is given for all atomic type X, so that $X^* = ([\![X]\!], \mathfrak{F}(X))$ is a finiteness space, and set $(A \to B)^* = A^* \Rightarrow B^*$, $(A \times B)^* = A^* \,\&\, B^*$ and $\top^* = \mathscr{T}$. Then, further assuming that, for all $a \in \mathfrak{C}_A$, $[\![a]\!] \in \mathfrak{F}(A^*)$, we obtain:

Theorem 3.2 (Finiteness). *If* $x_1 : A_1, \dots, x_n : A_n \vdash s : A$ *then* $[\![s]\!]_{x_1:A_1,\dots,x_n:A_n} \in \mathfrak{F}(A_1^* \Rightarrow \cdots \Rightarrow A_n^* \Rightarrow A^*)$.

Proof. This is a straightforward consequence of the fact that the cartesian closed structure of Fin is given by the same morphisms as in Rel. A direct proof is also possible, by induction on typing derivations. □

Examples. Pure typed λ-calculi are those with no additional constant or conversion rule: fix a set $\mathfrak{A}$ of atomic types, and write $\Lambda_0^{\mathfrak{A}}$ for the calculus where $\mathfrak{C}_A = \emptyset$ for all A, and $s \simeq t$ iff $s \simeq_0 t$. This is the most basic case and we have just shown that Rel and Fin model $\simeq_0$. Be aware that if we introduce no atomic type, then the semantics is actually trivial: in $\Lambda_0^{\emptyset}$, all types and terms are interpreted by $\emptyset$.

By contrast, we can consider the internal language $\Lambda_{\underline{\mathrm{Rel}}}$ of Rel in which all relations can be described: fix $\mathfrak{A}$ as the collection of all sets (or a fixed set of sets) and $\mathfrak{C}_A = \mathfrak{P}([\![A]\!])$. Then set $s \simeq_{\underline{\mathrm{Rel}}} t$ iff $[\![s]\!]_\Gamma = [\![t]\!]_\Gamma$, for any suitable Γ. The point in defining such a monstrous language is to enable very natural notations for relations: in general, we will identify closed terms in $\Lambda_{\underline{\mathrm{Rel}}}$ with the relations they denote in the empty context. For instance, we write $id_A = \lambda x x$ with x of type A; and if $f \in \underline{\mathrm{Rel}}(A,B)$ and $g \in \underline{\mathrm{Rel}}(B,C)$, we have $g \circ f = \lambda x (g\,(f\,x))$. Similarly, the internal language $\Lambda_{\underline{\mathrm{Fin}}}$ of Fin, where $\mathfrak{A}$ is the collection of all finiteness spaces and $\mathfrak{C}_A = \mathfrak{F}(A^*)$, allows to denote conveniently all finitary relations.

The main contribution of the present paper is to establish that Fin models Gödel's system T, which can be presented in various ways. The iterator version of system T is the typed λ-calculus with an atomic type Nat of natural numbers, and constants O of type Nat, S of type $\mathsf{Nat} \to \mathsf{Nat}$ and for all type A, I_A of type $\mathsf{Nat} \to (A \to A) \to A \to A$ and subject to the following additional conversions: $\mathsf{I}\,\mathsf{O}\,u\,v > v$ and $\mathsf{I}\,(\mathsf{S}\,t)\,u\,v > u\,(\mathsf{I}\,t\,u\,v)$ (we will in general omit the type subscript of such parametered constants). The recursor variant is similar, but the iterator is replaced with R_A of type $\mathsf{Nat} \to (\mathsf{Nat} \to A \to A) \to A \to A$ subject to conversions $\mathsf{R}\,\mathsf{O}\,u\,v > v$ and $\mathsf{R}\,(\mathsf{S}\,t)\,u\,v > u\,t\,(\mathsf{R}\,t\,u\,v)$. Those systems allow to represent exactly the same functions on the set of natural numbers, where the number n is denoted by $\mathsf{S}^n\,\mathsf{O}$: this is the consequence of a normalization theorem (see [8]). In fact, we can define a recursor using iteration and products with the standard encoding $\mathsf{rec} = \lambda x \lambda y \lambda z\, \pi_l\,(\mathsf{I}\,x\,(\lambda w \,\langle y\,(\pi_r\,w)\,(\pi_l\,w), \mathsf{S}\,(\pi_r\,w)\rangle)\,\langle z, \mathsf{O}\rangle)$, and we get $\mathsf{rec}\,(\mathsf{S}^n\,\mathsf{O})uv \simeq \mathsf{R}\,(\mathsf{S}^n\,\mathsf{O})uv$: the idea is to reconstruct the integer argument on the fly. But this encoding is valid only for ground terms of type Nat: $\mathsf{rec}\,(\mathsf{S}\,t)\,u\,v \simeq u\,t\,(\mathsf{rec}\,t\,u\,v)$ holds only if we suppose t is of the form $\mathsf{S}^n\,\mathsf{O}$, or reduces to such a term. By contrast, the encoding of the iterator by $\mathsf{iter} = \lambda x \lambda y \lambda z\,(\mathsf{R}\,x\,(\lambda x'\,y)\,z)$ is extensionally valid: $\mathsf{iter}\,\mathsf{O}\,u\,v \simeq v$ and $\mathsf{iter}\,(\mathsf{S}\,t)\,u\,v \simeq u\,(\mathsf{iter}\,t\,u\,v)$ for all t, u, v.

The fact that one direction of the encoding holds only on ground terms indicate that the algorithmic properties of both systems may differ. And these differences will appear in the semantics (see the final section). Also, recall the discussion in our introduction: the tail recursive variant of iterator, J subject to $\mathsf{J}\,(\mathsf{S}\,t)\,u\,v > \mathsf{J}\,t\,u\,(u\,v)$, uses its integer argument linearly. This enabled Ehrhard to define a semantics of iteration, with $\mathsf{Nat}^* = \mathscr{N} = (\mathbf{N}, \mathfrak{P}_\mathrm{f}(\mathbf{N}))$, $[\![\mathsf{O}]\!] = \mathscr{O} = \{0\}$ and $[\![\mathsf{S}]\!] = \mathscr{S} = \{([n], n+1);\ n \in \mathbf{N}\}$. Such an interpretation of natural numbers, however, fails to provide a semantics of I or R, in Rel or Fin.

Lemma 3.1. *Assume* $[\![\mathsf{Nat}]\!] = |\mathscr{N}|$, $[\![\mathsf{O}]\!] = \mathscr{O}$ *and* $[\![\mathsf{S}]\!] = \mathscr{S}$, *and let A be any type such that* $[\![A]\!] \neq \emptyset$. *Then there is no* $\mathscr{I}_A \subseteq [\![\mathsf{Nat} \to (A \to A) \to A \to A]\!]$ *such that, setting* $[\![\mathsf{I}_A]\!] = \mathscr{I}_A$, *we obtain* $[\![\mathsf{I}\,\mathsf{O}\,u\,v]\!]_\Gamma = [\![v]\!]_\Gamma$ *and* $[\![\mathsf{I}\,(\mathsf{S}\,t)\,u\,v]\!]_\Gamma = [\![u\,(\mathsf{I}\,t\,u\,v)]\!]_\Gamma$ *as soon as* $\Gamma \vdash t : \mathsf{Nat}$, $\Gamma \vdash u : A \to A$ *and* $\Gamma \vdash v : A$.

Proof. By contradiction, assume the above equations hold. By the second equation and Theorem 3.1, $\llbracket \mathsf{I}\,(\mathsf{S}x)\ (\lambda z'y)\ z\rrbracket = \llbracket y\rrbracket$, and thus $x^{[]} : \mathsf{Nat}, y^{[\alpha]} : A, z^{[]} : A \vdash \mathsf{I}\,(\mathsf{S}x)\ (\lambda z'y)\ z^{\alpha} : A$. Inversing the rules of Figure 2, we obtain that $([]\,,[([]\,,\alpha)]\,,[]\,,\alpha) \in \llbracket \mathsf{I}\rrbracket$ and then $([([]\,,\alpha)]\,,[]\,,\alpha) \in \llbracket \mathsf{I}\,\mathsf{O}\rrbracket$. Since $\llbracket A\rrbracket \neq \emptyset$, this contradicts the fact that, by the first equation: $\llbracket \mathsf{I}\,\mathsf{O}\rrbracket = \llbracket \lambda y\lambda z(\mathsf{I}\,\mathsf{O}\,yz)\rrbracket = \llbracket \lambda y\lambda z z\rrbracket = \{([]\,,[\alpha]\,,\alpha);\ \alpha \in \llbracket A\rrbracket\}$. □

4 A Finitary Relational Interpretation of Primitive Recursion

Lazy natural numbers. That $x^{[]} : \mathsf{Nat}, y^{[\alpha]} : A, z^{[]} : A \vdash \mathsf{I}\,(\mathsf{S}x)\ (\lambda z'y)\ z^{\alpha} : A$ implies $([]\,,[([]\,,\alpha)]\,,[]\,,\alpha) \in \llbracket \mathsf{I}\rrbracket$ holds because $\llbracket \mathsf{S}\rrbracket = \mathscr{S}$ is linear, hence strict: this reflects the general fact that, if $s \in \underline{\mathrm{Rel}}(A,B)$ contains no $([]\,,\beta)$ then, for all $t \in \underline{\mathrm{Rel}}(B,C)$, $([]\,,\gamma)$ in $t \circ s$ iff $([]\,,\gamma) \in t$. Such a phenomenon was also noted by Girard in his interpretation of system T in coherence spaces [8]. His evidence that there was no interpretation of the iteration operator using the linear successor relied on a coherence argument. The previous lemma is stronger: it holds in any web based model as soon as the interpretation of successor is strict.

In short, strict morphisms cannot produce anything *ex nihilo*; but the successor of any natural number should be marked as non-zero, for the iterator to distiguish between both cases. Hence the successor should be affine: similarly to Girard's solution, we will interpret Nat by so-called *lazy* natural numbers. Let $\mathscr{N}_l = (|\mathscr{N}_l|\,,\mathfrak{P}_{\mathrm{f}}(|\mathscr{N}_l|))$ be such that $|\mathscr{N}_l| = \mathbf{N} \cup \mathbf{N}^{>}$, where $\mathbf{N}^{>}$ is just a disjoint copy of $\mathbf{N}$. The elements of $\mathbf{N}^{>}$ are denoted by $k^{>}$, for $k \in \mathbf{N}$: $k^{>}$ represents a partial number, not fully determined but *strictly greater than k*. If $\nu \in |\mathscr{N}_l|$, we define ν^{+} as $k+1$ if $\nu = k$ and $(k+1)^{>}$ if $\nu = k^{>}$. Then we set $\mathscr{S}_l = \{([]\,,0^{>})\} \cup \{([\nu]\,,\nu^{+})\}$, which is affine. Notice that $\mathscr{O} \in \mathfrak{F}(\mathscr{N}_l)$ and $\mathscr{S}_l \in \mathfrak{F}(\mathscr{N}_l \Rightarrow \mathscr{N}_l)$.

Fixpoints. For all finiteness space $\mathscr{A}$, write $Rec[\mathscr{A}] = \mathscr{N}_l \Rightarrow (\mathscr{N}_l \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}$. We want to introduce a recursion operator $\mathscr{R}_{\mathscr{A}} \in \mathfrak{F}(Rec[\mathscr{A}])$ intuitively subject to the following definition: $\mathscr{R}\,t\,u\,v =$ match t with $\left\{\begin{array}{ccc} \mathsf{O} & \mapsto & v \\ \mathsf{S}t' & \mapsto & ut'\,(\mathscr{R}t'uv) \end{array}\right.$. This definition is recursive, and a natural means to obtain such an operator is as the fixpoint of $\mathscr{S}tep = \lambda\mathscr{X}\,\lambda x\lambda y\lambda z\left(\text{match } x \text{ with } \left\{\begin{array}{ccc} \mathsf{O} & \mapsto & z \\ \mathsf{S}x' & \mapsto & yx'\,(\mathscr{X}\,x'yz) \end{array}\right.\right)$. The cartesian closed category $\underline{\mathrm{Rel}}$ is cpo-enriched, the order on morphisms being inclusion. Hence it has fixpoints at all types: for all set A and $f \in \underline{\mathrm{Rel}}(A,A)$, the least fixpoint of f is $\bigcup_{k\geq 0} f^k\,\emptyset$, which is an increasing union. The least fixpoint operator is itself definable as the supremum of its approximants, $\mathscr{F}ix_A = \bigcup_{k\geq 0}\mathscr{F}ix_A^{(k)}$, where $\mathscr{F}ix_A^{(0)} = \emptyset$ and $\mathscr{F}ix_A^{(k+1)} = \lambda f\left(f\left(\mathscr{F}ix_A^{(k)}\,f\right)\right)$, more explicitly $\mathscr{F}ix_A^{(k+1)} = \left\{([([\alpha_1,\ldots,\alpha_n]\,,\alpha)] + \sum_{i=1}^{n}\overline{\varphi}_i,\alpha)\,;\ \forall i,\ (\overline{\varphi}_i,\alpha_i) \in \mathscr{F}ix_A^{(k)}\right\}$. Notice that these approximants are finitary: if $\mathscr{A}$ is a finiteness space then, for all k, $\mathscr{F}ix_{\mathscr{A}}^{(k)} = \mathscr{F}ix_{|\mathscr{A}|}^{(k)} \in \mathfrak{F}((\mathscr{A} \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A})$. The fixpoint, however, is not finitary in general: for instance $\mathscr{F}ix\,\mathscr{S}_l = \mathbf{N}^{>} \notin \mathfrak{F}(\mathscr{N}_l)$ hence $\mathscr{F}ix_{\mathscr{N}_l} \notin \mathfrak{F}((\mathscr{N}_l \Rightarrow \mathscr{N}_l) \Rightarrow \mathscr{N}_l)$. So we proceed in two steps: we first introduce the finitary approximants $\mathscr{R}_{\mathscr{A}}^{(k)} \in \mathfrak{F}(Rec[\mathscr{A}])$ by $\mathscr{R}_{\mathscr{A}}^{(k)} = \mathscr{S}tep_{\mathscr{A}}^{k}\,\emptyset$, then we prove $\mathscr{R}_{\mathscr{A}} = \bigcup_{k\geq 0}\mathscr{R}_{\mathscr{A}}^{(k)} \in \mathfrak{F}(Rec[\mathscr{A}])$.

Pattern matching on lazy natural numbers. We introduce a finitary operator $\mathscr{C}ase$, intuitively defined as: $\mathscr{C}ase\,t\,u\,v =$ match t with $\left\{\begin{array}{ccc} \mathsf{O} & \mapsto & v \\ \mathsf{S}t' & \mapsto & ut' \end{array}\right.$. More formally:

Definition 4.1. *If $\overline{\nu} = [\nu_1,\ldots,\nu_k] \in |\mathscr{N}_l|^{!}$, we write $\overline{\nu}^{+} = \left[\nu_1^{+},\ldots,\nu_n^{+}\right]$. Then for all set A, let $\mathscr{C}ase_A = \{([0]\,,[]\,,[\alpha]\,,\alpha);\ \alpha \in A\} \cup \left\{([0^{>}] + \overline{\nu}^{+},[(\overline{\nu},\alpha)]\,,[]\,,\alpha);\ \overline{\nu} \in |\mathscr{N}_l|^{!} \wedge \alpha \in A\right\}$.*

Lemma 4.1. *Pattern matching is finitary:* $\mathscr{C}ase_{\mathscr{A}} = \mathscr{C}ase_{|\mathscr{A}|} \in \mathfrak{F}(\mathscr{N}_l \Rightarrow (\mathscr{N}_l \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A})$*. Moreover,* $y : \mathscr{N}_l \Rightarrow \mathscr{A}, z : \mathscr{A} \vdash \mathscr{C}ase\, \mathscr{O}\, y z \simeq z : \mathscr{A}$ *and* $x : \mathscr{N}_l, y : \mathscr{N}_l \Rightarrow \mathscr{A}, z : \mathscr{A} \vdash \mathscr{C}ase\,(\mathscr{S}_l x) y z \simeq y x : \mathscr{A}$*.*

Proof. That the equations hold is a routine exercise. To prove $\mathscr{C}ase$ is finitary, we check the definition of $\mathfrak{F}(- \Rightarrow -)$. For the first direction: for all $n \in \mathfrak{F}(\mathscr{N}_l)$, $\mathscr{C}ase\, n \subseteq \{([\,], [\alpha], \alpha);\ \alpha \in |\mathscr{A}|\} \cup \left\{([(\overline{\nu}, \alpha)], [\,], \alpha);\ \overline{\nu}^{+} \in n^{!} \wedge \alpha \in |\mathscr{A}|\right\}$; hence, setting $n' = \{\nu;\ \nu^{+} \in n\} \in \mathfrak{F}(\mathscr{N}_l)$, we obtain $\mathscr{C}ase\, n \subseteq (\lambda y \lambda z z) \cup (\lambda y \lambda z (y n'))$, and we conclude since the union of two finitary subsets is finitary. In the reverse direction, we prove that, for all $\gamma \in |(\mathscr{N}_l \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}|$, setting $N' = \mathscr{C}ase^{\perp} \cdot \{\gamma\}$, $n^{!} \cap N'$ is finite; this is immediate because N' has at most one element. □

A recursor in $\underline{\mathrm{Rel}}$**.** We introduce the relation $\mathscr{R}$ as the fixpoint of $\mathscr{S}tep$.

Definition 4.2. *Fix a set A. Let* $\mathscr{S}tep_A = \lambda \mathscr{X}\, \lambda x \lambda y \lambda z (\mathscr{C}ase_A\, x (\lambda x' (y x' (\mathscr{X} x' y z))) z)$*. and, for all* $k \in \mathbf{N}$*, let* $\mathscr{R}_A^{(k)} = \mathscr{S}tep_A^{k}\, \emptyset$*. Then we define* $\mathscr{R}_A = \bigcup_{k \geq 0} \mathscr{R}_A^{(k)}$*, and fix* $[\![\mathsf{R}]\!] = \mathscr{R}$*.*

Lemma 4.2. *For all finiteness space* $\mathscr{A}$*,* $\mathscr{S}tep_{\mathscr{A}} = \mathscr{S}tep_{|\mathscr{A}|} \in \mathfrak{F}(Rec[\mathscr{A}] \Rightarrow Rec[\mathscr{A}])$ *and, for all k,* $\mathscr{R}_{\mathscr{A}}^{(k)} = \mathscr{R}_{|\mathscr{A}|}^{(k)} \in \mathfrak{F}(Rec[\mathscr{A}])$*. Moreover, we have:* $\mathscr{R}_{\mathscr{A}}^{(0)} = \emptyset$ *and* $\mathscr{R}_{\mathscr{A}}^{(k+1)} = \{([0], [\,], [\alpha], \alpha);\ \alpha \in |\mathscr{A}|\} \cup \left\{\left([0^{>}] + \sum_{i=0}^{n} \overline{\nu}_i^{+}, [(\overline{\nu}_0, [\alpha_1, \ldots, \alpha_n], \alpha)] + \sum_{i=1}^{n} \overline{\varphi}_i, \sum_{i=1}^{n} \overline{\alpha}_i, \alpha\right);\ \forall i,\ (\overline{\nu}_i, \overline{\varphi}_i, \overline{\alpha}_i, \alpha_i) \in \mathscr{R}_{\mathscr{A}}^{(k)}\right\}$*.*

Proof. The finiteness of the approximants follows from Theorem 3.2. The explicit description of $\mathscr{R}_{\mathscr{A}}^{(k)}$ is a direct application of the definition of the relational semantics. □

Theorem 4.3 (Correctness)**.** *For all suitable* Γ *and* Δ*,* $[\![\mathsf{R}\,\mathsf{O}\, y z]\!]_{\Gamma} = [\![z]\!]_{\Gamma}$ *and* $[\![\mathsf{R}\,(\mathsf{S} x)\, y z]\!]_{\Delta} = [\![y x (\mathsf{R} x y z)]\!]_{\Delta}$*.*

Proof. This follows directly from Lemma 4.1 and the fact that $\mathscr{R} = \mathscr{S}tep\, \mathscr{R}$. □

Finiteness. It only remains to prove $\mathscr{R}$ is finitary. Following the definition of $(- \Rightarrow -)$, we proceed in two steps: the image of a finitary subset of $\mathscr{N}_l$ is finitary; conversely, the preimage of a singleton is "anti-finitary".

Definition 4.4. *If* $\overline{\alpha} = [\alpha_1, \ldots, \alpha_k] \in a^{!}$*, we denote the support of* $\overline{\alpha}$ *by* $\mathrm{Supp}(\overline{\alpha}) = \{\alpha_1, \ldots, \alpha_k\} \subseteq a$*, and the size of* $\overline{\alpha}$ *by* $\#(\overline{\alpha}) = k$*. If* $n \in \mathfrak{F}(\mathscr{N}_l)$*, we set* $\max(n) = \max\{k;\ k \in n \vee k^{>} \in n\}$*, with the convention* $\max(\emptyset) = 0$*. Then if* $\overline{\nu} \in |\mathscr{N}_l|^{!}$ *we set* $\max(\overline{\nu}) = \max(\mathrm{Supp}(\nu))$*, and if* $\overline{n} \subseteq n^{!}$ *for some* $n \in \mathfrak{F}(\mathscr{N}_l)$*,* $\max(\overline{n}) = \max(\bigcup_{\overline{\nu} \in \overline{n}} \mathrm{Supp}(\overline{\nu}))$*.*

Lemma 4.3. *For all* $\gamma = (\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathscr{R}_{\mathscr{A}}$*,* $\gamma \in \mathscr{R}_{\mathscr{A}}^{(\max(\overline{\nu})+1)}$*.*

Proof. By induction on $\max(\overline{\nu})$, using Lemma 4.2. □

Lemma 4.4. *If* $n \in \mathfrak{F}(\mathscr{N}_l)$*, then* $\mathscr{R}_{\mathscr{A}}\, n \in \mathfrak{F}((\mathscr{N}_l \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A})$*.*

Proof. The previous Lemma entails $\mathscr{R}_{\mathscr{A}}\, n = \mathscr{R}_{\mathscr{A}}^{(\max(n)+1)}\, n$. We conclude recalling that $\mathscr{R}_{\mathscr{A}}^{(\max(n)+1)}\, n \in \mathfrak{F}((\mathscr{N}_l \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A})$, because $\mathscr{R}_{\mathscr{A}}^{(\max(n)+1)} \in Rec[\mathscr{A}]$. □

Definition 4.5. *For all* $\overline{\varphi} = [(\overline{\nu}_1, \overline{\alpha}_1, \alpha_1), \ldots, (\overline{\nu}_k, \overline{\alpha}_k, \alpha_k)] \in |\mathscr{N}_l \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}|^{!}$*, let* $\#\#(\overline{\varphi}) = \sum_{j=1}^{k} \#(\overline{\nu}_j)$*.*

Lemma 4.5. *If* $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathscr{R}_{\mathscr{A}}$*, then* $\#(\overline{\nu}) = \#(\overline{\alpha}) + \#(\overline{\varphi}) + \#\#(\overline{\varphi})$*.*

Proof. Using Lemma 4.2, the result is proved for all $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathscr{R}_{\mathscr{A}}^{(k)}$, by induction on k. □

Theorem 4.6 (The recursion operator is finitary)**.** $\mathscr{R}_{\mathscr{A}} \in \mathfrak{F}(Rec[\mathscr{A}])$*.*

Proof. By Lemma 4.4, we are left to prove that, for all $n \in \mathfrak{F}(\mathscr{N}_l)$ and $\gamma \in |(\mathscr{N}_l \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}) \Rightarrow \mathscr{A} \Rightarrow \mathscr{A}|$, $N = n^! \cap \left(\mathscr{R}^\perp \cdot \{\gamma\}\right)$ is finite. But by Lemma 4.5,

$$N \subseteq \left\{\overline{\nu} \in |\mathscr{N}_l|^!;\ \#(\overline{\nu}) = \#(\overline{\alpha}) + \#(\overline{\varphi}) + \#\#(\overline{\varphi}) \wedge \max(\overline{\nu}) \leq \max(n)\right\}$$

which is finite. □

Remark 4.7. *We keep calling $\mathscr{R}$ "the" recursion operator, but notice such an operator is not unique in* $\underline{\text{Rel}}$ *or* $\underline{\text{Fin}}$*: let* $\mathscr{C}ase'_A = \{([0,0],[],[\alpha],\alpha);\ \alpha \in A\} \cup \left\{\left([0^>] + \overline{\nu}^+, [(\overline{\nu},\alpha)], [], \alpha\right);\ \overline{\nu} \in |\mathscr{N}_l|^! \wedge \alpha \in A\right\}$, *for instance; this variant of matching operator behaves exactly like* $\mathscr{C}ase$*, and one can reproduce our construction of the recursor based on that.*

5 About Iteration

We have just provided a semantics of system T with recursor. Now let $\mathscr{I}_A = \lambda x \lambda y \lambda z\, (\mathscr{R}_A\, x\, (\lambda x'\, y)\, z)$ for all set A. By Theorem 4.6, $\mathscr{I}_{\mathscr{A}} = \mathscr{I}_{|\mathscr{A}|} \in \mathfrak{F}(\mathit{Iter}[\mathscr{A}])$. Moreover, by Theorem 4.3 this defines an iteration operator and we obtain that the triple $(|\mathscr{N}_l|, \mathscr{O}, \mathscr{S}_l)$, resp. $(\mathscr{N}_l, \mathscr{O}, \mathscr{S}_l)$, is a weak natural number object [12, 9] in the cartesian closed category $\underline{\text{Rel}}$, resp. $\underline{\text{Fin}}$.

We now develop a semantic argument demonstrating how recursion is stricly stronger than iteration. One distinctive feature of both models is non-uniformity: if $a, a' \in \mathfrak{F}(\mathscr{A})$ then $a \cup a' \in \mathfrak{F}(\mathscr{A})$; and in the construction of $a^!$, there is no restriction on the elements of the multisets we consider. It is very different from the setting of coherence spaces for instance. But we can show the iterator only considers uniform sets of lazy numbers, in the following sense: if $k \in \mathbf{N}$, we define $\underline{k} = \mathscr{S}_l^k\, \mathscr{O} = \{l^>;\ l < k\} \cup \{k\} \in \mathfrak{F}(\mathscr{N}_l)$; we say $n \subseteq |\mathscr{N}_l|$ is *uniform* if $n \subseteq \underline{k}$ for some k. Notice that, in the coherence space of lazy natural numbers used by Girard in [8] to interpret system T, the sets $\underline{k}$ are the finite maximal cliques: coherence is given by $k \,\Xi\, l$ iff $k = l$, $k \,\Xi\, l^>$ iff $k > l$ and $k^> \,\Xi\, l^>$ for all k, l. The only infinite maximal clique is $\mathbf{N}^>$ (recall this is the fixpoint of $\mathscr{S}_l$). We prove $\mathscr{I}$ considers only uniform sets of lazy numbers.

For all k, let $\mathscr{I}^{(k)}_{\mathscr{A}} = \lambda x \lambda y \lambda z \left(\mathscr{R}^{(k)}_{\mathscr{A}}\, x\, (\lambda x'\, y)\, z\right)$. Then let $\mathscr{S}tage^{(0)}_{\mathscr{A}} = \{([0],[],[\alpha],\alpha);\ \alpha \in |\mathscr{A}|\}$; $\mathscr{S}tage^{(1)}_{\mathscr{A}} = \{([0^>],[([],\alpha)],[],\alpha);\ \alpha \in |\mathscr{A}|\}$; and, for all $k > 0$, $\mathscr{S}tage^{(k+1)}_{\mathscr{A}} = \mathscr{I}^{(k+1)}_{\mathscr{A}} \setminus \mathscr{I}^{(k)}_{\mathscr{A}}$. One can check that $\mathscr{I}_{\mathscr{A}} = \bigcup_{k \geq 0} \mathscr{S}tage^{(k)}_{\mathscr{A}}$.

Lemma 5.1. *If* $\mathscr{A} \neq \mathscr{T}$ *then, for all* $k \in \mathbf{N}$, $\bigcup\left\{\mathrm{Supp}(\overline{\nu});\ \exists(\overline{\varphi}, \overline{\alpha}, \alpha),\ (\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathscr{S}tage^{(k)}_{\mathscr{A}}\right\} = \underline{k}$.

Proof. The inclusion $\subseteq$ is easy by induction on $\underline{k}$. For $\supseteq$, consider $\lambda x \lambda z \left(\mathscr{I}^{(k)}\, x\, (\lambda z'\, z') z\right)$. □

As a consequence, for all $(\overline{\nu}, \overline{\varphi}, \overline{\alpha}, \alpha) \in \mathscr{I}$, $\mathrm{Supp}(\overline{\nu})$ is uniform. Of course, no such property holds for $\mathscr{R}$, because $\mathscr{R}^{(1)}_{\mathscr{A}} \supseteq \left\{\left([0^>] + \overline{\nu}^+, [(\overline{\nu}, [], \alpha)], [], \alpha\right);\ \alpha \in |\mathscr{A}| \wedge \overline{\nu} \in |\mathscr{N}_l|^!\right\}$. An immediate generalization is that no recursor can be derived from $\mathscr{I}$: the interpretation of any recursor on the natural number object $(\mathscr{N}_l, \mathscr{O}, \mathscr{S}_l)$ necessarily contains elements of the above form.

Acknowledgements

The present work stems from a discussion with Thomas Ehrhard. It also greatly benefited from many working sessions in the company of Christine Tasson, to whom I am most grateful.

References

[1] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Not enough points is enough. In J. Duparc and T. Henzinger, eds., *Proc. of 21st Int. Wksh. on Computer Science Logic, CSL 2007 (Lausanne, Sept. 2007)*, v. 4646 of *Lect. Notes in Comput. Sci.*, pp. 298–312. Springer, 2007.

[2] T. Ehrhard. Finiteness spaces. *Math. Struct. in Comput. Sci.*, 15(4):615–646, 2005.

[3] T. Ehrhard and O. Laurent. Interpreting a finitary pi-calculus in differential interaction nets. In L. Caires and V. T. Vasconcelos, eds., *Proc. of 18th Int. Conf. on Concurrency Theory, CONCUR 2007 (Lisbon, Sept. 2007)*, v. 4703 of *Lect. Notes in Comput. Sci.*, pp. 333–348. Springer, 2007.

[4] T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1–3):1–41, 2003.

[5] T. Ehrhard and L. Regnier. Differential interaction nets. *Electron. Notes in Theor. Comput. Sci.*, 123:35–74, 2005.

[6] T. Ehrhard and L. Regnier. Böhm trees, Krivine's machine and the Taylor expansion of λ-terms. In A. Beckmann et al., eds., *Proc. of 2nd Conf. on Computability in Europe, CiE 2006 (Swansea, June/July 2006)*, v. 3988 of *Lect. Notes in Comput. Science*, pp. 186–197. Springer, 2006.

[7] J.-Y. Girard. Normal functors, power series and lambda-calculus. *Ann. of Pure and Appl. Log.*, 37(2):129–177, 1988.

[8] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, v. 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1989.

[9] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*, v. 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge Univ. Press, 1988.

[10] M. Pagani and C. Tasson. The inverse Taylor expansion problem in linear logic. In *Proc. of 24th Ann. IEEE Symp. on Logic in Computer Science (Los Angeles, CA, Aug. 2009)*, IEEE CS Press, to appear.

[11] C. Tasson. Algebraic totality, towards completeness. In P.-L. Curien, ed., *Proc of 9th Int. Conf. on Typed Lambda Calculi and Applications, TLCA 2009 (Brasilia, July 2009)*, v. 5608 of *Lect. Notes in Comput. Sci.*, pp. 325–340. Springer, 2009.

[12] M.-F. Thibault. Pre-recursive categories. *J. of Pure and Appl. Alg.*, 24:79–93, 1982.

[13] P. Tranquilli. Intuitionistic differential nets and lambda-calculus. *Theor. Comput. Sci*, to appear.

[14] L. Vaux. Differential linear logic and polarization. In P.-L. Curien, ed., *Proc of 9th Int. Conf. on Typed Lambda Calculi and Applications, TLCA 2009 (Brasilia, July 2009)*, v. 5608 of *Lect. Notes in Comput. Sci.*, pp. 371–385. Springer, 2009.

Author index