EPTCS 435

Proceedings of the

Twelfth Workshop on

Fixed Points in Computer Science

Naples, Italy, 19-20th February 2024

Edited by: Alexis Saurin

Published: 4th November 2025
DOI: 10.4204/EPTCS.435
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Preface

Alexis Saurin
IRIF, Université Paris Cité, CNRS & INRIA Picube
alexis.saurin@irif.fr

This volume contains the proceedings of the Twelfth International Workshop on Fixed Points in Computer Science (FICS 2024) which took place on February 19th and 20th, 2024 in Naples, Italy, as a satellite event of the conference Computer Science Logic (CSL 2024) in the Center of Congress of the Università degli Studi di Napoli Federico II.

Fixed points play a fundamental role in several areas of computer science. They are used to justify (co)recursive definitions and associated reasoning techniques. The construction and properties of fixed points have been investigated in many different settings such as: design and implementation of programming languages and proof assistants, logics, verification or databases. The aim of this workshop is to provide 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.

For the 12th edition of the workshop, a great variety of topics have been presented witnessing notably research on the μ-calculus and modal logics, on dependent type theory and mechanisation of proofs, on denotational and categorical semantics, on non-wellfounded proof theory as well as on infinitary λ-calculi, notably using nominal techniques and using fixed-points to solve game-theoretic and automata-theoretic problems. This diversity is reflected in the contributions published in these post-proceedings.

This year, FICS accepted two categories of contributions, short abstracts for presentation only and extended abstracts to be presented and then published in this volume of post-proceedings. Every submission was evaluated by three reviewers (mostly the PC members[1] as well as some external reviewers). We received a total of 18 submissions: the PC selected 17 for presentation at the workshop of which there were 8 extended abstract to be published in this volume. Apart from presentations of the accepted abstracts, we are delighted that FICS 2024 featured two invited talks by Anupam Das (University of Birmingham) on *the computational strength of circular proofs* and by Barbara König (University of Duisburg-Essen) on *Approximating Fixpoints of Approximated Functions*. König's invited talk was joint with CSL conference.

For reference, we include below the program of the workshop (including unpublished contributions) and the abstracts of the invited talks.

Monday 19 February

- Session 1 (8:50 – 10:20, Chair: Alexis Saurin)
  - 08:50 – 09:00: Workshop opening.

[1]The program committee was constituted of Zena Ariola, Abhishek De, Zeinab Galal, Guilhem Jaber, Ekaterina Komendantskaya, Denis Kuperberg, Martin Lange, Christine Paulin, Daniela Petrisan, Alexis Saurin, Thomas Studer, Tarmo Uustalu, and Yde Venema.

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. iii–vi, doi:10.4204/EPTCS.435.0

- 09:00 – 9:50: Invited talk by Anupam Das. On the Computational Strength of Circular Proofs

  The last 10 years has seen a surge of renewed interest in non-wellfounded proofs, from modal and predicate logics to more structural proof theoretic endeavours. A particularly fascinating direction, pioneered by the French school, is to construe circular proofs as a typing discipline for functional programs, à la Curry-Howard. Circular typing has a notable advantage against traditional (co)recursors: the operational semantics are more explicit, closer to low-level machine models with loops while nonetheless enjoying excellent metalogical properties. After foundational work of Santocanale, Baelde and Clairambault in the 2000s, the subject was considerably advanced in the 2010s by several researchers in the Paris circle.
  A recurring question running through these works has been to gauge the precise computational strength of circular proofs. The obvious problem here is the non-constructive character of totality/soundness arguments for non-wellfounded proofs, a barrier to traditional approaches. In this talk I will survey a recent line of work building up a general methodology for answering such questions via a metamathematics. In particular, we manage to pin down the precise computational strengths of a circular version of system T and its extension by least and greatest fixed points. This talk is partly based on joint work with Gianluca Curzi.
- 9:50 – 10:20: Bahareh Afshari, Giacomo Barlucchi and Graham E. Leigh. The Limit of Recursion in State-based Systems
- 10:20 – 10:45: Coffee break

- Session 2 (10:45 – 12:45, Chair: Anupam Das)
  - 10:45 – 11:15: Gianluca Curzi, Matteo Acclavio and Giulio Guerrieri. Non-Uniform Polynomial Time and Non-Wellfounded Parsimonious Proofs
  - 11:15 – 11:45: Borja Sierra-Miranda. Cyclic Proofs for iGL via Corecursion
  - 11:45 – 12:15: Bahareh Afshari and Johannes Kloibhofer. Cut Elimination for Cyclic Proofs: A Case Study in Temporal Logic
  - 12:15 – 12:45: Esaïe Bauer and Alexis Saurin. Cut-Elimination for the Circular Modal μ-Calculus: the Benefits of Linearity
  - 12:45 – 14:00: Lunch

- Session 3 (14:00 – 16:00, Chair: Benedikt Ahrens)
  - 14:00 – 14:30: Zeinab Galal and Jean-Simon Pacaud Lemay. Combining Fixpoint and Differentiation Theory
  - 14:30 – 15:00: Luigi Santocanale and Gregory Chichery. Lifting Final Coalgebras and Initial Algebras, a Reconstruction
  - 15:00 – 15:30: Paige Randall North and Maximilien Peroux. Coinductive Control of Inductive Data Types
  - 15:30 – 16:00: Tadeusz Litak. Ruitenburg's Theorem Mechanized and Contextualized
  - 16:00 – 16:30: Coffee break

- Session 4 (16:30 – 18:30, Chair: Zeinab Galal)
  - 16:30 – 17:00: Farzad Jafarrahmani and Noam Zeilberger. A Fibrational Characterization for Unicity of Solutions to Generalized Context-Free Systems

- 17:00 – 17:30: Rémy Cerda. Nominal Algebraic-Coalgebraic Data Types, with Applications to Infinitary λ-Calculi
- 17:30 – 18:00: Ralph Matthes, Kobe Wullaert and Benedikt Ahrens. Substitution for Non-Wellfounded Syntax with Binders through Monoidal Categories
- 18:00 – 18:30: Mohamed Hamza Bandukara and Nikos Tzevelekos. Nominal Logics for Fresh Register Automata

Tuesday 20 February

- Session 5 (09:30 – 10:30, Chair: Bahareh Afshari)
  - 09:30 – 10:30 Invited talk (joint with CSL'24) by Barbara König. Approximating Fixpoints of Approximated Functions

    There is a large body of work on fixpoint theorems, guaranteeing the existence of fixpoints for certain functions and providing methods for computing them. This includes for instance Banach's fixpoint theorem, the well-known result by Knaster-Tarski that is frequently employed in computer science and Kleene iteration.
    It is less clear how to compute fixpoints if the function whose (least) fixpoint we are interested in is not known exactly, but can only be obtained by a sequence of subsequently better approximations. This scenario occurs for instance in the context of reinforcement learning, where the probabilities of a Markov decision process (MDP) – for which one wants to learn a strategy – are unknown and can only be sampled. There are several solutions to this problem where the fixpoint computation (for determining the value vector and the optimal strategy) and the exploration of the model are interleaved. However, these methods work only well for discounted MDPs, that is in the contractive setting, but not for general MDPs, that is for non-expansive functions.
    After describing and motivating the problem, we will in particular concentrate on the non-expansive case. There are many interesting systems who value vectors can be obtained by determining the fixpoints of non-expansive functions. Other than contractive functions, they do not guarantee uniqueness of the fixpoint, making it more difficult to approximate the least fixpoint by methods other than Kleene iteration. And also Kleene iteration fails if the function under consideration is only approximated.
    We hence describe a dampened Mann iteration scheme for (higher-dimensional) functions on the reals that converges to the least fixpoint from everywhere. This scheme can also be adapted to functions that are approximated, under certain conditions.
    We will in particular study the case of MDPs and consider a related problem that arises when performing model-checking for quantitative mu-calculi, which involves the computation of nested fixpoints.
    This is joint work with Paolo Baldan, Sebastian Gurke, Tommaso Padoan and Florian Wittbold.
  - 10:30 – 11:00: Coffee break
- Session 6 (11:00 – 13:05, Chair: Abhishek De)
  - 11:00 – 11:30: Daniel Hausmann. Faster Game Solving by Fixpoint Acceleration

- 11:30 – 12:00: Florian Bruse, David Kronenberger and Martin Lange. Characterizing the Exponential-Space Hierarchy Via Partial Fixpoints
- 12:00 – 12:30: Leonardo Pacheco. The μ-calculus' Alternation Hierarchy is Strict over Non-Trivial Fusion Logics
- 12:30 – 13:00: Daniel Hausmann, Nir Piterman, Irmak Saglam and Anne-Kathrin Schmuck. Fixpoints for Fair Parity/$\perp$Games
- 13:00 – 13:05: Workshop conclusion

More details about the 2024 edition of the workshop (abstracts of the talks, slides or supplementary material) as well as FICS workshop series can be found online at `https://www.irif.fr/users/saurin/fics2024/index.html`.

The editor thanks all authors who submitted contributions to FICS 2024, the program committee members for their work in selecting the papers presented of the workshop, as well as Florian Bruse and Johannes Kloibhofer for their help in the scientific evaluation of the submissions. A special thank to Ralph Matthes for chairing FICS steering committee and Denis Kuperberg for advices on the organization of the workshop. Many thanks to Anupam Das and Barbara König for accepting our invitation. Finally, we would like to express our deep gratitude to CSL 2024 for the local organization and to EACSL and ANR ("Agence Nationale de la Recherche", France) for funding FICS 2024.

Alexis Saurin,
FICS 2024 PC Chair

The Limit of Recursion in State-based Systems*

Bahareh Afshari Giacomo Barlucchi Graham E. Leigh
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg, Gothenburg, Sweden
bahareh.afshari@gu.se giacomo.barlucchi@gu.se graham.leigh@gu.se

We prove that ω^2 strictly bounds the iterations required for modal definable functions to reach a fixed point across all countable structures. The result corrects and extends the previously claimed result by the first and third authors on closure ordinals of the alternation-free μ-calculus in [3]. The new approach sees a reincarnation of Kozen's well-annotations, devised for showing the finite model property for the modal μ-calculus. We develop a theory of 'conservative' well-annotations where minimality of annotations is guaranteed, and isolate parts of the structure that locally determine the closure ordinal of relevant formulas. This adoption of well-annotations enables a direct and clear pumping process that rules out closure ordinals between ω^2 and the limit of countability.

1 Introduction

State-based systems and processes lay at the heart of computer science. Abstractly, they are no more than directed graphs, also known as Kripke frames, with states as vertices and state transitions as edges. Taking a transition from one state to another can model a step in computation and, doing so recursively, singles out computation paths through the system. To specify and verify properties of computation, in a fully abstract manner, temporal logics offer an elegant framework.

Syntactically simple and algorithmically rich, temporal logics have been heavily studied. Amongst them the modal μ-calculus holds a special place, providing a level of abstraction that is mathematically appealing while computationally well-behaved. This logic not only subsumes well known temporal logics (LTL, CTL, PDL, as well as extensions such as μLTL and CTL*), it can also be enriched to capture, for example, probabilistic properties [5, 21, 24], hyperproperties [14], higher-dimensional properties [28] (see also [20]), and properties of higher-order recursive schemes [27, 4]. In other words, μ-calculus is a cornerstone in the mosaic of logics in computer science.

Modal μ-calculus is the extension of basic modal logic with least (μ) and greatest (ν) fixed point operators. Over a Kripke frame $\mathscr{S}$, the formula $\mu x\,\varphi(x)$ is interpreted as the least fixed point of the induced monotone function $f\colon U \mapsto \varphi(U)$ which maps a set of states U (in $\mathscr{S}$) to the denotation of $\varphi(x)$ modulo interpretation of x as U. This fixed point can be obtained as the limit of transfinite iterations of f. Starting with the empty set, applications of f give rise to an increasing sequence of sets of states,

$$\emptyset \subseteq f(\emptyset) \subseteq f(f(\emptyset)) \subseteq \cdots \subseteq f^{\alpha}(\emptyset) \subseteq f^{\alpha+1}(\emptyset) \subseteq \cdots$$

which necessarily stabilises at some ordinal: $f^{\kappa+1}(\emptyset) = f^{\kappa}(\emptyset)$. The least such κ is the closure ordinal of f in $\mathscr{S}$. One way to define a notion of *closure ordinal* for a formula $\mu x\,\varphi$ is as the supremum of closure ordinals of the induced function across all frames.

In this paper we study closure ordinals of the Σ-fragment: formulas generated from closed μ-calculus formulas and variables through the logical and modal operators, and the least fixed point operator μ.

*This work was supported by the Knut and Alice Wallenberg Foundation [2020.0199], Swedish Research Council [2017-05111] and Dutch Research Council [OCENW.M20.048]

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 1–12, doi:10.4204/EPTCS.435.1

From an algebraic perspective the fragment corresponds to functions definable in the modal algebra with (definable) parameters. A more general class, amounting to the full calculus, is to admit arbitrary definable functions, including those defined through co-recursion.

There are several problems concerning closure ordinals which, to date, have only partial solutions. As there are countably many formulas of μ-calculus there are countable ordinals that are not closure ordinals. So, *which* ordinals are closure ordinals? Aside from *existence* is the question of *decidability*: Is there an algorithm that decides whether any given formula has a closure ordinal? And not least is the question of *limitedness*: Can a non-trivial limit on closure ordinals be determined?

1.1 Related work

Closure ordinals have been considered only by a handful of authors. Most notable is the Czarnecki formulas [6], simple formulas in the Σ-fragment demonstrating that every ordinal below ω^2 is a closure ordinal. Czarnecki's formulas indicate a connection between syntactic and semantic complexity that was generalised in [3]: consider formulas of the form $\mu x.\varphi$ with φ given by

$$\varphi = (p_1 \wedge \Box q_1 \wedge \bigcirc_1 x) \vee (p_2 \wedge \Box q_2 \wedge \bigcirc_2 x) \vee \cdots \vee (p_n \wedge \Box q_n \wedge \bigcirc_n x) \vee \Box \bot$$

where p_i and q_i are conjunctions of literals and $\bigcirc_i \in \{\Diamond, \Box\}$ for each i. It is not difficult to prove that if such a formula has a closure ordinal α, then $\alpha < \omega.(n+1)$. In [3], the authors also provide a tableaux-based characterisation for the closure ordinals of the alternation-free μ-calculus. In [17], Kretz proves that every valid Σ_1-formula in the one-variable fragment, which includes any valid formula of the form above, has finite closure ordinal.

Fontaine [9] (see also [10]) carries out a study of closure ordinals of the continuous μ-calculus, that is the fragment constituting formulas $\mu x.\varphi(x)$ where $\varphi(x)$ is continuous with respect to x in the Scott topology on the powerset algebra. It is shown the $\{\Box, \vee\}$-free fragment of μ-calculus characterises the continuous μ-calculus establishing that closure ordinals are obtained in at most ω iterations. Fontaine and Venema provide syntactic characterisations of several other semantic properties in [11]. Gouveia and Santocanale [12] study κ-continuity for κ an infinite (regular) cardinal, and prove a generalisation of the aforementioned results regarding existence: any ordinal obtainable from 0, 1, ω, and ω_1 (least uncountable ordinal) by the binary ordinal sum operation is the closure ordinal of a μ-calculus formula.

One may argue that the concept of closure ordinal and questions posed about it stand somewhat remote to other investigations concerning μ-calculus. But there is strong evidence that this not so. One intriguing connection, pointed out by Skrzypczak [30], is to the descriptive complexity of Büchi languages. Each Büchi automaton can be associated a rank below ω_1, measuring the complexity of the automaton against input trees. It is shown that the rank of an automaton $\mathscr{B}$ is strictly below ω_1 if and only if the language of $\mathscr{B}$ is Borel, and strictly below ω^2 if and only if the language is weak monadic second order definable. Skrzypczak proposes that the pumping arguments central to deducing bounds on closure ordinals may be used to tackle questions of definability (and decidability) of non-deterministic languages, the so-called *gap properties* for Büchi languages (see e.g. [29, 26]).

Milanese [22] studies closure ordinals over bidirectional models and shows that every ordinal below ω^ω is a closure ordinal (see also [23]). This result was observed independently in [1] as part of a study of ω-branching proof systems for the two-way μ-calculus. Again, such results add weight to the claim that closure ordinals are entwined in many topics concerning μ-calculi.

1.2 Contribution

We prove that ω^2 is a strict upper bound on the closure ordinals of the Σ-fragment of the modal μ-calculus. This reproves and extends the claims in [3] concerning closure ordinals of the alternation-free fragment. There are two critical errors in [3], both in the original proof of the 'pumping' Lemma 3.18.[1] While the errors can be fixed (see unpublished notes [2]) it is at the cost of a weaker result and a more technically involved argument that appeals, in particular, to the closure ordinals of valid Σ_1-formulas.

The approach presented here develops a theory of ordinal annotations that simplifies the conceptual framework compared to [2] and lays the groundwork for future extensions to the full calculus. At its base is the notion of well-annotations, employed by Kozen to establish the finite model property for μ-calculus [16]. We refine the concept by imposing constraints on the annotating ordinals so that the existence of such a 'conservative' well-annotation corresponds to the existence of certain closure ordinals. The theory of well-annotations becomes more tractable by also restricting the underlying syntax. Rather than the traditional syntax of the modal μ-calculus, we consider formulas constructed via a single modal operator – related to the 'cover' modality of [15] – and present them as modal equation systems in conjunctive form. It is shown that both expressivity and bounds on closure ordinals are preserved through this syntactic preprocessing. The central argument involves a pumping lemma for well-annotations. Assuming the existence of a sufficiently 'large' conservative well-annotation, a transfinite series of substitutions shows it possible to obtain a conservative well-annotation corresponding in size to an arbitrary countable ordinal, thereby refuting the existence of closure ordinals equal or greater than ω^2.

2 Modal μ-calculus

We adopt a unimodal presentation of modal logic eschewing the usual unary modal operators $\Box$ and $\Diamond$ for a single modality ∇ that takes finitely many formulas as arguments. The intended interpretation of $\nabla\Gamma$ in terms of $\Box/\Diamond$ syntax is $\bigvee_{\gamma\in\Gamma}\Box\gamma\vee\Diamond\bigwedge\Gamma$. This modality is the classical dual of the 'cover' modality originally introduced by Janin and Walukiewicz [15] and has proved especially well suited for investigating the modal and co-algebraic logics [15, 32, 18, 19]. The formulas of the modal μ-calculus, denoted $\mathscr{L}_\mu$, are those generated by the following grammar.

$$\varphi := p \mid \overline{p} \mid x \mid \bigvee\Gamma \mid \bigwedge\Gamma \mid \nabla\Gamma \mid \mu x\varphi \mid \nu x\varphi$$
$$\Gamma := \emptyset \mid \Gamma\cup\{\varphi\}$$

where x and p range over, respectively, a set Var of *variables* and Prop of *propositional constants*. Note, negation is not included as a logical connective, except for propositional constants, expressed by the atoms $\overline{p}$ above. Non-variable atoms, namely propositional constants and their negations, are called *literals*, the set of which is denoted Lit.

We utilise abbreviations $\bot := \bigvee\emptyset$ and $\top := \bigwedge\emptyset$, and represent binary conjunction and disjunction via, respectively, $\varphi_0\wedge\varphi_1 := \bigwedge\{\varphi_0,\varphi_1\}$ and $\varphi_0\vee\varphi_1 := \bigvee\{\varphi_0,\varphi_1\}$. With the intended interpretation of ∇, the two unary modalities $\Box$ and $\Diamond$ are recovered by $\Box\varphi := \nabla\{\varphi,\bot\}$ and $\Diamond\varphi := \nabla\{\varphi\}\wedge\nabla\emptyset$.

Free and bound variables are defined per usual. A formula with no free variable occurrences is called *closed* and we write $\mathscr{L}_\mu^-$ for the set of closed formulas. For a set of formulas F, the *quantifier-free formulas over* F, denoted $\mathsf{QF}[F]$, is the closure of $F\cup\mathsf{Lit}$ under the logical connectives and ∇.

Formulas are interpreted with respect to Kripke frames. A *frame* is a tuple $\mathscr{S} = (S,R,\Lambda)$ comprising a non-empty set S of *states*, a binary *accessibility* relation $R\subseteq S\times S$ and a *labelling* function $\Lambda\colon \mathsf{Prop}\to$

[1] The authors are indebted to Michał Skrzypczak and Igor Walukiewicz for identifying one of the errors in [3].

$\mathscr{P}(S)$ from propositional constants to sets of states. A frame is often identified with its set of states. For a frame (S,R,Λ) and $s \in S$, we write $R[s]$ for the set of successors of s, namely, $\{t \in S \mid (s,t) \in R\}$. Given a formula, a frame $\mathscr{S} = (S,R,\Lambda)$ and a valuation function $\mathscr{V}\colon \mathsf{Var} \to \mathscr{P}(S)$, the denotation of φ in $\mathscr{S}$ relative to $\mathscr{V}$ is the set $\|\varphi\|^{\mathscr{S}}_{\mathscr{V}}$ defined by

$$\|x\|^{\mathscr{S}}_{\mathscr{V}} = \mathscr{V}(x) \qquad \|\bigwedge\Gamma\|^{\mathscr{S}}_{\mathscr{V}} = \bigcap\{\, \|\gamma\|^{\mathscr{S}}_{\mathscr{V}} \mid \gamma \in \Gamma \,\} \qquad \|\nu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} = \bigcup\{\, U \subseteq S \mid U \subseteq \|\varphi\|^{\mathscr{S}}_{\mathscr{V}[x\mapsto U]} \,\}$$
$$\|p\|^{\mathscr{S}}_{\mathscr{V}} = \Lambda(p) \qquad \|\bigvee\Gamma\|^{\mathscr{S}}_{\mathscr{V}} = \bigcup\{\, \|\gamma\|^{\mathscr{S}}_{\mathscr{V}} \mid \gamma \in \Gamma \,\} \qquad \|\mu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} = \bigcap\{\, U \subseteq S \mid \|\varphi\|^{\mathscr{S}}_{\mathscr{V}[x\mapsto U]} \subseteq U \,\}$$
$$\|\overline{p}\|^{\mathscr{S}}_{\mathscr{V}} = S \setminus \Lambda(p) \qquad \|\nabla\Gamma\|^{\mathscr{S}}_{\mathscr{V}} = \nabla\{\, \|\varphi\|^{\mathscr{S}}_{\mathscr{V}} \mid \varphi \in \Gamma \,\}$$

where $\mathscr{V}[x \mapsto U]$ expresses the valuation that maps x to U and otherwise agrees with $\mathscr{V}$. The function $\nabla\colon \mathscr{P}(\mathscr{P}(S)) \to \mathscr{P}(S)$ is specified by

$$\nabla\mathscr{U} := \{\, v \in S \mid R[v] \subseteq U \text{ for some } U \in \mathscr{U} \,\} \cup \{\, v \in S \mid \textstyle\bigcap \mathscr{U} \cap R[v] \neq \emptyset \,\}.$$

2.1 Equational formulas

We will be working with the Σ-fragment of $\mathscr{L}_\mu$ which, loosely speaking, consists of formulas wherein the external μ-quantifiers do not bind variables in the scope of other quantifiers. More precisely, the Σ-*fragment* is the closure of $\mathscr{L}^-_\mu \cup \mathsf{Lit} \cup \mathsf{Var}$ under the logical connectives, ∇-modality and the μ-quantification. We refer to formulas of the Σ-fragment as Σ-*formulas*.

For the analysis we adopt a representation of Σ-formulas based on *modal equation systems*. A modal equation system (m.e.s.) consists of a finite set of equations between variables and quantifier-free formulas, accompanied by a 'priority' order on variables. We spare the general definition of m.e.s. (for which the reader can consult, e.g., [7, sec. 8.3.4]) and focus on a formulation corresponding to Σ-formulas. In particular, in our set-up there is no order imposed on variables, and equations relate each variable to a quantifier-free formula over $\mathscr{L}^-_\mu$ and variables, that is, $\mathsf{QF}[\mathscr{L}^-_\mu \cup \mathsf{Var}]$.

Definition 1. An *equation system* over $\mathscr{L}^-_\mu$ is a tuple (X,E) where $X \subseteq \mathsf{Var}$ is a finite set of variables and $E\colon X \to \mathsf{QF}[X \cup \mathscr{L}^-_\mu]$ is such that all variables in $E(x)$ are in the scope of a modality. An *equational formula* (over $\mathscr{L}^-_\mu$) is a triple (X,x_0,E) where (X,E) is an equation system (over $\mathscr{L}^-_\mu$) and $x_0 \in X$ is a distinguished variable called the *initial variable*.

The intended semantics of an equational formula is the denotation of the initial variable relative to the system's equations taken under a least fixed point reading. The formal semantics is most easily given through *approximations*:

Definition 2 (Approximations)**.** Fix an equation system (X,E) and frame $\mathscr{S}$. For each ordinal α define a valuation $\mathscr{V}^\alpha$ by

$$\mathscr{V}^\alpha(x) = \begin{cases} \bigcup_{\beta<\alpha} \|E(x)\|^{\mathscr{S}}_{\mathscr{V}^\beta}, & x \in X, \\ \emptyset, & x \in \mathsf{Var} \setminus X. \end{cases}$$

For each formula $\psi \in \mathsf{QF}[X \cup \mathscr{L}^-_\mu]$, the α-th *approximation* of ψ (relative to (X,E)), also referred to as the *denotation* of ψ^α, is $\|\psi^\alpha\|^{\mathscr{S}} := \|\psi\|^{\mathscr{S}}_{\mathscr{V}^\alpha}$. The denotation of the equational formula $\varphi = (X,x,E)$ in $\mathscr{S}$ is defined as $\|\varphi\|^{\mathscr{S}} := \bigcup_{\alpha<\omega_1} \|x^\alpha\|^{\mathscr{S}}$.

That every Σ-formula is equivalent to an equational formula over $\mathscr{L}^-_\mu$ can be shown via a simple translation between the two representations that replaces the 'external' μ-operators by equations and vice-versa. Henceforth, we identify Σ-formulas and equational formulas.

We will utilise a special form of equational systems/formulas that facilitates the desired pumping argument while staying faithful to both expressivity and closure ordinals within the Σ-fragment.

Definition 3 (Conjunctive system)**.** An equation system (X,E) is said to be *conjunctive* if for every $x \in X$, the formula $E(x)$ is of the form $\bigwedge_{i<k}(\bigvee \Gamma_i \vee \nabla Y_i)$ for some $\Gamma_i \subseteq \mathscr{L}_\mu^-$ and $Y_i \subseteq X$. An equational formula over a conjunctive system is called a *conjunctive formula*.

One obvious constraint is that in the syntax above it is not possible to express $\bot$ as the empty disjunction. Instead, $\bot$ is expressed as the conjunctive equation $z \mapsto \Diamond z$ where, recall, $\Diamond z = \nabla\{z\} \wedge \nabla\emptyset$. Note also that, in a conjunctive equation, the modal depth is trivial and a conjunct may contain at most one ∇-modality. That the resulting fragment is as expressive as the Σ-fragment is essentially the dual of Janin and Walukiewicz' 'disjunctive normal form' theorem [15]. Less obvious is the preservation of closure ordinals which will be addressed in the next section (see Theorem 7).

It is worth highlighting that what we have called 'conjunctive' here is most correctly the 'conjunctive Σ-fragment'. Since we only work with the Σ-fragment in this article we opt for the shorter name convention.

We use the following adaptation of the standard Fischer–Ladner closure of formulas [8] to equation systems. The *closure* of an equation system (X,E) is the smallest set $\mathsf{Clos}(X,E) \subseteq \mathscr{L}_\mu$ satisfying (1) $E(X) \subseteq \mathsf{Clos}(X,E)$; (2) if $\bigcirc\Gamma \in \mathsf{Clos}(X,E)$ for $\bigcirc \in \{\bigwedge, \bigvee, \nabla\}$ then $\Gamma \subseteq \mathsf{Clos}(X,E)$; and (3) if $\sigma x\,\psi \in \mathsf{Clos}(X,E)$ for $\sigma \in \{\mu,\nu\}$ then $\psi(\sigma x\,\psi/x) \in \mathsf{Clos}(X,E)$ where $\psi(\chi/x)$ denotes the result of substituting χ for free occurrences of x in ψ, avoiding variable capture. The size of the equation system (X,E), written $|X,E|$, is the cardinality of its closure.

2.2 Closure ordinals

As remarked, an $\mathscr{L}_\mu$-formula $\varphi(x)$ considered over a frame $\mathscr{S}$ induces a monotone function on the powerset lattice $(\mathscr{P}(S),\subseteq)$ mapping a set of states $U \subseteq \mathscr{S}$ to $\|\varphi\|^{\mathscr{S}}_{\mathscr{V}[x\mapsto U]}$. One may give an approximation semantics for $\mathscr{L}_\mu$ by iterating this function into the transfinite, where Ω denotes the class of ordinals:

$$\|\mu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} = \bigcup_{\kappa\in\Omega} \|\mu^\kappa x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} \quad \text{where} \quad \|\mu^\alpha x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} := \bigcup_{\beta<\alpha} \|\varphi\|^{\mathscr{S}}_{\mathscr{V}[x\mapsto\|\mu^\beta x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}}]} \tag{1}$$

The 'formula' $\mu^\kappa x\varphi$ expresses the κ-iteration of the function f starting on $\emptyset$. In particular, for every $s \in \|\mu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}}$ there is some $\kappa \in \Omega$ s.t. $s \in \|\mu^\kappa x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}}$. If $\mathscr{S}$ is a countable frame, cardinality considerations show that $\|\mu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} = \|\mu^\kappa x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}}$ for some $\kappa < \omega_1$. Thus, for each closed $\mu x\,\varphi$, there exists $\kappa \leq \omega_1$ such that $\|\mu x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}} = \|\mu^\kappa x\,\varphi\|^{\mathscr{S}}_{\mathscr{V}}$ for *every* countable frame $\mathscr{S}$ and valuation $\mathscr{V}$.

It is essentially these approximations that provide the semantics of equational formulas in the previous section though there is a noteworthy difference: in Definition 2 the ordinal annotation adopted consists of a single ordinal number 'counting' multiple variables. Implicit in (1) is what is known as the *signature*, an n-tuple of ordinals keeping record of the iterations of each μ operator in $\mu x\,\varphi$. In the context of an equation system (X,E) a signature is an assignment of an ordinal to each variable in X. Assuming a fixed enumeration $x_0,\ldots,x_n$ of X, a signature is a sequence of ordinals $\alpha_0\cdots\alpha_n$ and the denotation of quantifier-free formulas over X relative to this signature uses α_i to interpret x_i: $\|x_i^{\alpha_0\cdots\alpha_n}\|^{\mathscr{S}} = \bigcup_{\beta<\alpha_i}\|E(x_i)^{\cdots\alpha_{i-1}\beta\,\alpha_{i+1}\cdots}\|^{\mathscr{S}}$. For a detailed definition and properties of signatures we refer the reader to [31, 25, 7]. The 'single approximation' notion of denotation in Definition 2, which counts each and every unfolding of equations, may appear a crude measure in comparison to the fine-grained specification that treats each equation independently. Signatures, however, also introduce complications of a 'book-keeping' nature while offering a level of detail that is not needed for characterising bounds on fixed point iterations as shown by the following lemma.

Lemma 4. *Let $\alpha_0,\ldots,\alpha_n < \omega_1$ and $\alpha = \sum_i \alpha_i$. For every equational formula ψ with variables over $x_0,\ldots,x_n$ and structure $\mathscr{S}$, $\|\psi^{\alpha_0\cdots\alpha_n}\|^{\mathscr{S}} \subseteq \|\psi^{\alpha}\|^{\mathscr{S}} \subseteq \|\psi^{\alpha\cdots\alpha}\|^{\mathscr{S}}$.*

Following the notion of ordinal approximation in Definition 2 we define closure ordinals of Σ-formulas as follows.

Definition 5 (Closure Ordinal)**.** Given a frame $\mathscr{S}$ and Σ-formula φ presented as an equational formula, *the closure ordinal of* φ *in* $\mathscr{S}$ is the least ordinal $\kappa = \mathsf{CO}_{\mathscr{S}}(\varphi)$ such that $\|\varphi^{\kappa}\|^{\mathscr{S}} = \|\varphi\|^{\mathscr{S}}$. The *closure ordinal* of φ, denoted $\mathsf{CO}(\varphi)$, is the least ordinal κ such that for all countable frames $\mathscr{S}$, $\mathsf{CO}_{\mathscr{S}}(\varphi) \leq \kappa$.

As the definition above restricts attention to countable frames, every formula has closure ordinal bounded by the first uncountable ordinal ω_1. Cardinality considerations show that not every countable ordinal is a closure ordinal. Yet it is open as to precisely which countable ordinals are closure ordinals. The following partial result was established by Czarnecki [6].

Proposition 6. *For every* $\alpha < \omega^2$ *there exists a* Σ*-formula* φ *such that* $\mathsf{CO}(\varphi) = \alpha$.

We end this section with a result on preservation of closure ordinals between equivalent formulas.

Theorem 7. *Let* φ *be a* Σ*-formula. There exists a conjunctive formula* φ_c *such that* $\mathsf{CO}(\varphi) \leq \mathsf{CO}(\varphi_c)$*, and* $\mathsf{CO}(\varphi) < \omega_1$ *implies* $\mathsf{CO}(\varphi_c) < \omega_1$.

The proof of Theorem 7 proceeds by converting each equational formula into an equivalent conjunctive one. This transformation is, in essence, determinisation of alternating parity tree automata [13]. Due to space reasons, the syntactic translation from arbitrary equational formulas to conjunctive ones is not shown here. We point out, however, that for the theorem we require that the conjunctive formula so obtained preserves existence of a countable closure ordinal, a property which is not invariant under mere logical equivalence (the reader can compare formulas $\mu x \top$ and $\mu x(\Box x \vee \nu y \Diamond y)$).

3 Conservative well-annotations

The notion of well-annotations is taken from [16] with minor changes to adapt to conjunctive equation systems. An annotated formula is a pair (φ, α), written φ^{α}, where φ is a formula and $\alpha < \omega_1$. For a set Θ of annotated formulas, Θ^{-} denotes the underlying formulas: $\Theta^{-} := \{\, \varphi \mid \varphi^{\alpha} \in \Theta \text{ for some } \alpha < \omega_1 \,\}$. For Γ a set of unannotated formulas, let $\Gamma^{\alpha} = \{\, \varphi^{\alpha} \mid \varphi \in \Gamma \,\}$. We utilise a relation $\preceq$ on sets of annotated formulas, defined by $\Theta \preceq \Xi$ iff for all $\varphi^{\alpha} \in \Xi$ there exists $\beta \leq \alpha$ such that $\varphi^{\beta} \in \Theta$.

Definition 8 (Well-annotation)**.** An *annotation* of a frame $\mathscr{S}$ for an equation system (X, E) is a function $\Theta\colon \mathscr{S} \to \mathscr{P}(\mathscr{L}_{\mu} \times \omega_1)$ associating to each state $s \in \mathscr{S}$ a set Θ_s of annotated formulas from $\mathsf{Clos}(X, E)$. A *well-annotation* of $\mathscr{S}$ for (X, E) is an annotation Θ such that for all $s \in \mathscr{S}$, φ, Γ and $\alpha < \omega_1$:

1. if $\varphi^{\alpha} \in \Theta_s$ and $\varphi \in \mathscr{L}_{\mu}^{-}$ then $s \in \|\varphi\|^{\mathscr{S}}$;
2. if $x^{\alpha} \in \Theta_s$ for $x \in X$, then $E(x)^{\beta} \in \Theta_s$ for some $\beta < \alpha$;
3. if $\bigvee \Gamma^{\alpha} \in \Theta_s$ then $\Gamma^{\beta} \cap \Theta_s \neq \emptyset$ for some $\beta \leq \alpha$;
4. if $\bigwedge \Gamma^{\alpha} \in \Theta_s$ then $\Theta_s \preceq \Gamma^{\alpha}$;
5. if $\nabla \Gamma^{\alpha} \in \Theta_s$, then one of the following properties holds
   (a) there exists $r \in R[s]$ such that $\Theta_r \preceq \Gamma^{\alpha}$,
   (b) there exists $\varphi \in \Gamma$ such that $\Theta_r \preceq \{\, \varphi^{\alpha} \,\}$ for all $r \in R[s]$.

Recall that quantified formulas in this setting are all in $\mathscr{L}_{\mu}^{-}$, hence they are considered in 1. When referring to well-annotations we omit explicit mention of the underlying frame and associated equation system if there is no cause for confusion. The relation $\preceq$ introduced above is extended to annotations in a pointwise manner. That is, for annotations Θ and Ξ of a frame $\mathscr{S}$, set $\Theta \preceq \Xi$ iff $\Theta_s \preceq \Xi_s$ for all $s \in \mathscr{S}$. In the following, $\mathscr{S}, s \vDash \Theta_s$ expresses that $s \in \|\varphi^{\alpha}\|^{\mathscr{S}}$ for every $\varphi^{\alpha} \in \Theta_s$.

Theorem 9. *Given an annotation Θ of $\mathscr{S}$,*

1. *If Θ is a well-annotation, then $\mathscr{S}, s \vDash \Theta_s$ for every $s \in \mathscr{S}$.*
2. *If $\mathscr{S}, s \vDash \Theta_s$ for all s, then there is a well-annotation Θ' of $\mathscr{S}$ such that $\Theta' \preceq \Theta$.*

Proof. See [16, Lemma 4.2]. □

We are interested in well-annotations that are $\preceq$-minimal for a given frame and equation system. We call these annotations *conservative*.

Definition 10 (Conservative well-annotation)**.** A well-annotation Θ of $\mathscr{S}$ is conservative if two conditions are met:

1. for every $s \in \mathscr{S}$ and φ there is at most one $\alpha < \omega_1$ such that $\varphi^\alpha \in \Theta_s$.
2. for every well-annotation Θ' of $\mathscr{S}$, $\Theta \preceq \Theta'$.

The existence of conservative well-annotations is guaranteed by

Proposition 11. *Let $\mathscr{S}$ be a frame, $r \in S$ and $x \in X$ such that $r \in \|x\|^{\mathscr{S}}$. There exists a conservative well-annotation Θ of $\mathscr{S}$ such that $x \in \Theta_r^-$.*

Proof. The desired annotation is given by $\varphi^\alpha \in \Theta_s$ iff $s \in \|\varphi\|^{\mathscr{S}}$ and α is least such that $s \in \|\varphi^\alpha\|^{\mathscr{S}}$. □

The following proposition provides the crucial link between conservative well-annotations and closure ordinals.

Proposition 12. *Suppose Θ is a conservative well-annotation of $\mathscr{S}$ and $\varphi^\alpha \in \Theta_s$ for some $s \in \mathscr{S}$. Then $\mathsf{CO}(\varphi) \geq \alpha$.*

Proof. Let Θ be a conservative well-annotation of $\mathscr{S}$ and $\varphi^\alpha \in \Theta_s$. Assume $\mathsf{CO}(\varphi) = \gamma < \alpha$. By Definition 5, $s \in \|\varphi^\alpha\|^{\mathscr{S}} = \|\varphi^\gamma\|^{\mathscr{S}}$. Consider the annotation $\hat{\Theta}$ given by $\hat{\Theta}_s = \Theta_s \cup \{\varphi^\gamma\}$ and $\hat{\Theta}_r = \Theta_r$ for any $r \neq s$. Theorem 9 implies that $\hat{\Theta}$ can be extended to a well-annotation Θ' of $\mathscr{S}$ satisfying $\Theta'_s \preceq \hat{\Theta}_s$. But then $\Theta \not\preceq \Theta'$ contradicting the assumption that Θ is conservative. We conclude that $\mathsf{CO}(\varphi) \geq \alpha$. □

Not every formula in a conservative well-annotation of φ^α plays a role in generating α as ordinal. From the large quantity of information provided by a conservative well-annotation, we want to be able to identify the part of the annotation that is relevant in determining the main ordinal, i.e., its *relevant part*. Specifically, for a formula φ^α that is in the relevant part of some Θ_s, the definition of relevant part guarantees that a new frame with φ annotated by some ordinal $> \alpha$ can be obtained, if it is possible to alter the initial frame in a way that (1) does not alter the formulas satisfied at the successors of s, and (2) increases the ordinal annotation of every relevant formula at a successor of s to some ordinal $> \alpha$.

For the following we introduce some notation concerning a well-annotation Θ of $\mathscr{S}$. Given the twofold condition on $\nabla\Gamma^\alpha \in \Theta_s$ in Definition 8, it is useful to identify the sets witnessing the two existential claims. For $s \in \mathscr{S}$ and $\Gamma \subseteq \mathscr{L}_\mu$, define

$$\Gamma^s_\Box := \{\, \varphi \in \Gamma \mid \varphi \in \bigcap_{t \in R[s]} \Theta_t^- \,\} \qquad\qquad \Gamma^s_\Diamond := \{\, t \in R[s] \mid \Gamma \subseteq \Theta_t^- \,\}.$$

For $\alpha > 0$, define $\mathsf{p}(\alpha)$ as the least ordinal $< \alpha$ such that $\alpha = \mathsf{p}(\alpha) + \omega^\eta$ for some η. Note also that η is uniquely determined and independent of the choice of $\mathsf{p}(\alpha)$.

Definition 13 (Relevant part)**.** Let Θ be a conservative well-annotation of $\mathscr{S}$ and Φ an annotation of the same frame. We call Φ a *relevant part* of Θ if for every $s \in \mathscr{S}$,

1. $\Phi_s \subseteq \Theta_s$;
2. if $x^\alpha \in \Phi_s$ then $E(x)^\beta \in \Phi_s$ where $\alpha = \beta + 1$;
3. if $\bigvee\Gamma^\alpha \in \Phi_s$ and $\alpha > 0$ then $\Gamma^\alpha \cap \Theta_s \subseteq \Phi_s$;
4. if $\bigwedge\Gamma^\alpha \in \Phi_s$ then $\chi^\alpha \in \Phi_s$ for exactly one $\chi \in \Gamma$;
5. if $\nabla\Gamma^\alpha \in \Phi_s$ and $\alpha > 0$ then:
   (a) for all $\eta < \alpha$ and $\varphi \in \Gamma^s_\Box$ there is a $r \in R[s]$ and $\beta > \eta$ s.t. $\varphi^\beta \in \Phi_r$,
   (b) $\Gamma \cap \Phi_r^- \neq \emptyset$ for every $r \in \Gamma^s_\Diamond$, and
   (c) for all $r \in R[s]$ if $\Phi_r \neq \emptyset$ then $y^\beta \in \Phi_r$ for some $y \in \Gamma$ and $\beta > \mathsf{p}(\alpha)$.

Formulas in Φ_s are referred to as *relevant formulas* at s. The final condition of the definition, 5c, has the role of ensuring that the formulas relevant at a successor state sit in the same ordinal 'neighbourhood'. Since $\mathsf{p}(\omega+1) = \omega$ and $\mathsf{p}(\omega.(k+1)) = \omega.k$, in these cases all continuations through a modality should be annotated by at least $\omega+1$ and $\omega.k$ respectively. In other words, viewing the sequences of formulas in the relevant part as a formula 'trace' through the well-annotation Θ, these traces are restricted in the size of their ordinal decrements. Notice, however, that the ordinal annotations along these relevant 'traces' need not be weakly decreasing. If for $r \in R[s]$ we have $\Gamma \subseteq \Theta_r^-$ then we require some $\psi^\beta \in \Gamma^\beta \cap \Theta_r$ to be marked as relevant even if $\beta > \alpha$. The reason for this requirement is that such a successor r, although not 'relevant' to witnessing the ordinal of $\nabla\Gamma$ in Θ_s may become 'relevant' after an attempt to force an increase in the ordinal annotation. It could be the case, for example, that for some $r \in R[s]$ we have $\Gamma^{\alpha+2} \subseteq \Theta_r$; if the annotation of all relevant formulas in $R[s] \setminus \{r\}$ is increased by, say, ω then without also increasing the annotation at r we find that $\nabla\Gamma^\alpha$ at s has increased only to $\nabla\Gamma^{\alpha+2}$.

In order to isolate sufficient conditions for undertaking a pumping of well-annotations, a further constraint can be placed on relevant parts to the effect that each path through the underlying frame carries at most one relevant trace of formulas. In the context of conjunctive formulas, this condition amounts to there being at most one relevant modal formula at each state.

Proposition 14. *Let $\varphi = (X, x, E)$ be a conjunctive formula. Let Θ be a conservative well-annotation and $x^\alpha \in \Theta_s$ for some $\alpha < \omega_1$ and state s. There exists a tree $\mathscr{T}$, a conservative well-annotation Θ' of $\mathscr{T}$ and a relevant part Φ satisfying:*

1. *$x^\alpha \in \Phi_r$ where r is the root of $\mathscr{T}$.*
2. *For every $t \in \mathscr{T}$, there is at most one Γ such that $\nabla\Gamma \in \Phi_t^-$.*

The crux of the argument is in ensuring that the requirements of Definition 13 can be met while marking at most one modal formula as relevant at each state. The restriction to conjunctive formulas makes condition 5 of the definition the only non-trivial case. Duplicating successor nodes enables the desired assignment of relevant formulas to successors.

4 Limits on closure ordinals

The argument showing that ω^2 is an upper bound on the closure ordinals of formulas in the Σ-fragment comprises two parts. First, the existence of a pumping procedure for sufficiently large conservative well-annotations is established. As a consequence, for every formula φ in the fragment there is a measure

N, related to the size of the formula, that determines an interval $[\omega.N, \omega^2)$ where the possibility of the closure ordinal of φ is excluded. In the second part, the interval is extended to all the (countable) ordinals above $\omega.N$ by proving that the consequences of the pumping method reach beyond ω^2. Combining the two parts, we obtain $\omega^2 = \sup_k \omega.k$ as an upper bound for the Σ-fragment.

A *path* in a frame $\mathscr{S} = (S, R, \Lambda)$ is a sequence of states $(s_i)_{i \leq k}$ such that $s_{i+1} \in R[s_i]$ for $i < k$. An infinite path through $\mathscr{S}$ is an infinite sequence $(s_i)_{i<\omega}$ such that every initial sequence is a path through $\mathscr{S}$. If $\mathscr{S}$ is a tree we use ρ to denote the root state. For the following let a conjunctive equation system (X, E) be fixed.

Definition 15. Given $\varphi, \Gamma \subseteq \mathsf{Clos}(X, E)$, define $O(\varphi, \Gamma)$ to be the supremum of ordinals $\kappa < \omega_1$ for which there exists a conservative well-annotation Θ of a tree such that $\Gamma = \Theta^-_\rho$ and $\varphi^\kappa \in \Theta_\rho$.

Definition 16 (Optimal annotation)**.** Given a state s in a conservative well annotation Θ, Θ_s is *optimal* with respect to a formula $\varphi^\alpha \in \Theta_s$ if $O(\varphi, \Theta_s) < \alpha + \omega$.

Since a key element of the argument in the proof of Lemma 20 will be the non-existence of optimal paths under certain conditions, we introduce the notion of repetition pair. As recognized by Lemma 18, repetition pairs are designed to entail non-optimality, given that they present candidates for additional pumping.

Definition 17 (Repetition pair)**.** Let Θ be a conservative well-annotation of $\mathscr{S}$ and Φ a relevant part of Θ. A state $s \in \mathscr{S}$ is a *limit state* of Φ if $\nabla\Gamma^\lambda \in \Phi_s$ for some Γ and limit ordinal λ. A pair of states (s, t) in $\mathscr{S}$ is a *repetition pair* if:

1. there is a path $(s_i)_{i \leq k}$ with $s = s_0$ and $t = s_k$,
2. $(\Theta^-_s, \Phi^-_s) = (\Theta^-_t, \Phi^-_t)$,
3. s and t are limit states and $\nabla\Gamma^\alpha \in \Phi_s$ and $\nabla\Gamma^\beta \in \Phi_t$ for some Γ and $\beta < \alpha$.

We refer to t as the *bud* and s as the *companion* of the repetition pair.

Call a path whose limit states are all optimal an optimal path.

Lemma 18. *On every optimal path there are no repetition pairs.*

Proof. Straightforward from the fact that the bud node is a non-optimal limit point by definition. □

Finally, the next proposition specifies sufficient conditions for the existence of repetition pairs.

Proposition 19. *Let $N = 2^{2|\varphi|} + 1$ and Θ a conservative well-annotation of $\mathscr{S}$ with respect to φ. Let $(s_i)_{i \leq k}$ be a path through $\mathscr{S}$ and $(\varphi_i, \alpha_i)_{i \leq k}$ a sequence of annotated formulas such that $\varphi_i^{\alpha_i} \in \Phi_{s_i}$ for each i. If $\alpha_0 \geq \omega.N$ and $\alpha_k = 0$, then some (s_i, s_j) is a repetition pair.*

Proof. Definition 13 ensures that on every sufficiently long path in which the relevant part remains non-empty, there are $2^{2|\varphi|}$ limit states with the corresponding limit ordinal strictly decreasing between states. As there are $2^{|\varphi|}$ subsets of $\mathsf{Clos}(\varphi)$ the existence of a repetition pair is immediate. □

Consider a state Θ_s in Θ with relevant part Φ_s such that $\nabla\Gamma^\lambda \in \Phi_s$. If $O(\nabla\Gamma, \Theta^-_s) \geq \lambda + \omega$, by definition there is a conservative well-annotation Θ' such that $\Theta^-_s = (\Theta'_\rho)^-$ and $\nabla\Gamma^\beta \in \Theta'_\rho$ for some $\beta \geq \lambda + \omega$. We call *pumping* of Θ_s the operation of replacing in Θ the branch rooted at Θ_s with the conservative well-annotation Θ'.

Lemma 20 (First pumping lemma)**.** *There exists $N < \omega$ such that for all $\Gamma \subseteq \mathsf{Clos}(X, E)$ and $x \in X$, if $O(x, \Gamma) < \omega^2$ then $O(x, \Gamma) \leq \omega.N$.*

Proof sketch. We prove the contrapositive statement, i.e., if $O(x,\Gamma) > \omega.N$ then $O(x,\Gamma) \geq \omega^2$ for suitable N. Let $N = 2^{2|\varphi|}+1$ and assume a formula (X,E) and a Θ such that $O(x,\Theta_\rho) = \kappa$ for some $\omega.N < \kappa < \omega^2$. We prove that the root of such Θ cannot be optimal, i.e. that $O(x,\Theta_\rho) > \kappa$, by showing that that would entail the existence of another conservative well-annotation Θ' with an optimal path and a repetition pair, contradicting Lemma 18. From the fact that Θ_ρ is assumed optimal, we proceed by pumping every successor $r \in R[\rho]$ that is not optimal (wrt the unique relevant $\nabla\Gamma_r$). Since by definition there must be at least one optimal $s \in R[\rho]$, we move to all optimal successors and repeat the pumping of their non-optimal successors. The conservative well-annotation Θ' obtained at the end of this process by definition has an optimal path that is strictly decreasing, hence fulfilling the conditions of Proposition 19, from which we obtain the contradiction with Lemma 18. It follows that $O(x,\Theta_\rho) = \kappa > \omega.N$ entails $\kappa \geq \omega^2$. □

The first pumping lemma eliminates ordinals sufficiently close to ω^2 as being closure ordinals of Σ-formulas of bounded size. In the rest of the argument we do the same for ordinals between ω^2 and ω_1.

Lemma 21 (Second pumping lemma)**.** *For all $\kappa < \omega_1$, $\Gamma \subseteq \mathsf{Clos}(X,E)$ and $x \in X$, if $O(x,\Gamma) \leq \kappa$ then $O(x,\Gamma) < \omega^2$.*

Proof sketch. The proof proceeds by transfinite induction on κ and is similar in spirit to the first pumping lemma. This second lemma doesn't rely directly on optimality because a greater generality is needed, but Lemma 20 serves as base case in the argument for every $\kappa \geq \omega^2$. Given a conservative well-annotation Θ with $x^\kappa \in \Theta_\rho$ for $\kappa \geq \omega^2$ a series of substitutions to the underlying tree induces a tree that satisfies x at the root and for which all well-annotations necessitate a strictly larger annotation of this variable. In the case that $\kappa > \omega^2$ is a successor ordinal, the substitutions can be performed directly to the successors of the root by appealing to the induction hypothesis. The case of a limit ordinal is more involved and requires identifying, via the relevant part of Θ, transfinitely many candidate states at which the induction hypothesis can be applied. After performing the substitutions, an infinite descent argument establishes a necessary increase in the ordinal annotation. □

Theorem 22. *A countable ordinal α is the closure ordinal of a formula in the Σ-fragment iff $\alpha < \omega^2$.*

Proof. One direction is provided by Proposition 6. For the other direction, let φ be a Σ-formula with $\mathsf{CO}(\varphi) < \omega_1$. By Theorem 7 we may assume φ is conjunctive. Thus, given x the initial variable of φ, we have $\mathsf{CO}(\varphi) = \alpha$ entails $O(x,\Delta) = \alpha$ for some $\Delta \subseteq \mathsf{Clos}(X,E)$, whence $\mathsf{CO}(\varphi) < \omega^2$ by Lemma 21. □

5 Conclusion

We have shown that the countable closure ordinals of formulas in the Σ-fragment are strictly bounded by ω^2. The result extends what was claimed in [3] and is obtained using a different method that circumvents the shortcomings of the approach taken there. The main ingredient introduced is a reworked version of well-annotation from [16], with which the focus is cast directly on transforming frames.

The machinery described in this article is for most part independent of the Σ-fragment to which they are applied. An immediate continuation of this work is, therefore, to examine the versatility of the tools for investigating closure ordinal of $\mu x\varphi$ where φ is any μ-calculus formula. Another research direction, suggested by some of the insights from Section 2.2, is to directly study the relation between closure ordinals and semantic equivalence, i.e., the syntactic operations which preserve closure ordinals.

References

[1] Bahareh Afshari, Gerhard Jäger & Graham E. Leigh (2019): *An Infinitary Treatment of Full Mu-Calculus*. In Rosalie Iemhoff, Michael Moortgat & Ruy J. G. B. de Queiroz, editors: *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, *Lecture Notes in Computer Science* 11541, Springer, pp. 17–34, doi:10.1007/978-3-662-59533-6_2.

[2] Bahareh Afshari & Graham E. Leigh: *Closure Ordinals: Revisions and Proofs*. Available at https://surfdrive.surf.nl/files/index.php/s/AWpStp9s2SynBno. Unpublished notes.

[3] Bahareh Afshari & Graham E. Leigh (2013): *On closure ordinals for the modal mu-calculus*. In Simona Ronchi Della Rocca, editor: *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, *LIPIcs* 23, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 30–44, doi:10.4230/LIPIcs.CSL.2013.30.

[4] Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong & Olivier Serre (2021): *Higher-order Recursion Schemes and Collapsible Pushdown Automata: Logical Properties*. *ACM Trans. Comput. Log.* 22(2), pp. 12:1–12:37, doi:10.1145/3452917.

[5] Pablo F. Castro, Cecilia Kilmurray & Nir Piterman (2015): *Tractable Probabilistic mu-Calculus That Expresses Probabilistic Temporal Logics*. In Ernst W. Mayr & Nicolas Ollinger, editors: *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, *LIPIcs* 30, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 211–223, doi:10.4230/LIPIcs.STACS.2015.211.

[6] Marek Czarnecki (2010): *How fast can the fixpoints in modal mu-calculus be reached*. *Fixed Points in Computer Science*, pp. 35–39.

[7] Stéphane Demri, Valentin Goranko & Martin Lange (2016): *The Modal Mu-Calculus*. In: *Temporal Logics in Computer Science: Finite-State Systems*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, pp. 271–328, doi:10.1017/CBO9781139236119.008.

[8] Michael J. Fischer & Richard E. Ladner (1979): *Propositional dynamic logic of regular programs*. *Journal of Computer and System Sciences* 18(2), pp. 194–211, doi:10.1016/0022-0000(79)90046-1.

[9] Gaëlle Fontaine (2008): *Continuous Fragment of the mu-Calculus*. In Michael Kaminski & Simone Martini, editors: *Computer Science Logic*, Springer Berlin Heidelberg, pp. 139–153, doi:10.1007/978-3-540-87531-4_12.

[10] Gaëlle Fontaine (2010): *Modal fixpoint logic: some model theoretic questions*. Ph.D. thesis, University of Amsterdam.

[11] Gaëlle Fontaine & Yde Venema (2018): *Some model theory for the modal mu-calculus: syntactic characterisations of semantic properties*. *Logical Methods in Computer Science; Volume 14*, p. Issue 1, doi:10.23638/LMCS-14(1:14)2018.

[12] Maria João Gouveia & Luigi Santocanale (2019): $\aleph_1$ *and the modal* μ*-calculus*. *Log. Methods Comput. Sci.* 15(4), doi:10.23638/LMCS-15(4:1)2019.

[13] Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer Berlin, Heidelberg, doi:10.1007/3-540-36387-4.

[14] Jens Oliver Gutsfeld, Markus Müller-Olm & Christoph Ohrem (2021): *Automata and fixpoints for asynchronous hyperproperties*. *Proc. ACM Program. Lang.* 5(POPL), pp. 1–29, doi:10.1145/3434319.

[15] David Janin & Igor Walukiewicz (1995): *Automata for the modal mu-calculus and related results*. In Jiří Wiedermann & Petr Hájek, editors: *Mathematical Foundations of Computer Science 1995*, *Lecture Notes in Computer Science* 969, Springer Berlin Heidelberg, pp. 552–562, doi:10.1007/3-540-60246-1_160.

[16] Dexter Kozen (1988): *A Finite Model Theorem for the Propositional mu-Calculus*. *Studia Logica: An International Journal for Symbolic Logic* 47(3), pp. 233–241, doi:10.1007/BF00370554.

[17] Mathis Kretz (2006): *Proof-theoretic aspects of modal logic with fixed points*. Ph.D. thesis, University of Bern.

[18] Clemens Kupke, Alexander Kurz & Yde Venema (2012): *Completeness for the coalgebraic cover modality*. *Log. Methods Comput. Sci.* 8(3), doi:10.2168/LMCS-8(3:2)2012.

[19] Clemens Kupke & Dirk Pattinson (2011): *Coalgebraic semantics of modal logics: An overview*. *Theor. Comput. Sci.* 412(38), pp. 5070–5094, doi:10.1016/j.tcs.2011.04.023.

[20] Martin Lange (2015): *The Arity Hierarchy in the Polyadic μ-Calculus*. In Ralph Matthes & Matteo Mio, editors: *Proceedings Tenth International Workshop on Fixed Points in Computer Science, FICS 2015, Berlin, Germany, September 11-12, 2015*, *EPTCS* 191, pp. 105–116, doi:10.4204/EPTCS.191.10.

[21] Wanwei Liu, Lei Song, Ji Wang & Lijun Zhang (2015): *A Simple Probabilistic Extension of Modal Mu-calculus*. In Qiang Yang & Michael J. Wooldridge, editors: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, AAAI Press, pp. 882–888, doi:10.48550/arXiv.1504.07737.

[22] Gian Carlo Milanese (2018): *An exploration of closure ordinals in the modal μ-calculus*. Master thesis, University of Amsterdam.

[23] Gian Carlo Milanese & Yde Venema (2019): *Closure Ordinals of the Two-Way Modal μ-Calculus*. In Rosalie Iemhoff, Michael Moortgat & Ruy J. G. B. de Queiroz, editors: *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, *Lecture Notes in Computer Science* 11541, Springer, pp. 498–515, doi:10.1007/978-3-662-59533-6_30.

[24] Matteo Mio & Alex Simpson (2017): *Łukasiewicz μ-calculus*. *Fundam. Informaticae* 150(3-4), pp. 317–346, doi:10.3233/FI-2017-1472.

[25] Damian Niwiński & Igor Walukiewicz (1996): *Games for the mu-calculus*. *Theoretical Computer Science* 163(1), pp. 99–116, doi:10.1016/0304-3975(95)00136-0.

[26] Damian Niwiński & Igor Walukiewicz (2003): *A gap property of deterministic tree languages*. *Theor. Comput. Sci.* 303(1), pp. 215–231, doi:10.1016/S0304-3975(02)00452-8.

[27] C.-H. Luke Ong (2006): *On Model-Checking Trees Generated by Higher-Order Recursion Schemes*. In: *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, IEEE Computer Society, pp. 81–90, doi:10.1109/LICS.2006.38.

[28] Martin Otto (1999): *Bisimulation-invariant PTIME and higher-dimensional μ-calculus*. *Theor. Comput. Sci.* 224(1-2), pp. 237–265, doi:10.1016/S0304-3975(98)00314-4.

[29] Michal Skrzypczak & Igor Walukiewicz (2016): *Deciding the Topological Complexity of Büchi Languages*. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani & Davide Sangiorgi, editors: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 55, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 99:1–99:13, doi:10.4230/LIPIcs.ICALP.2016.99.

[30] Michał Skrzypczak (2014): *Descriptive set theoretic methods in automata theory*. PhD thesis, University of Warsaw, doi:10.1007/978-3-662-52947-8.

[31] Robert S. Streett & E. Allen Emerson (1989): *An automata theoretic decision procedure for the propositional mu-calculus*. *Information and Computation* 81(3), pp. 249–264, doi:10.1016/0890-5401(89)90031-X.

[32] Igor Walukiewicz (2000): *Completeness of Kozen's Axiomatisation of the Propositional mu-Calculus*. *Information and Computation* 157(1), pp. 142–182, doi:10.1006/inco.1999.2836.

Cyclic Proofs for iGL via Corecursion

Borja Sierra Miranda*
Logic and Theory Group, University of Bern
borja.sierra@unibe.ch

1 Introduction

Cyclic proof theory studies proofs where cycles are allowed (see [1]). This is useful for developing proof theory for logics with fixpoint operators: cycles can be used to represent the unfolding of a fixpoint. However, this cyclic character is not unique to such explicit fixpoints. For example, modal logics whose frames have a Noetherian (conversely wellfounded) condition, such as GL (Gödel-Löb logic, see [8],[6]), S4Grz (Grzegorczyk logic, see [7]) and K4Grz (see [10]) also have cyclic proof systems.

Particularly, in [8], Shamkanov introduces a non-wellfounded and a cyclic sequent system (GL). He proves the equivalence of these two systems with an acyclic finite system via proof translations. In order to go from the finite system to the non-wellfounded system he defines the translation by corecursion.

In [6], Iemhoff generalized the work of Shamkanov studying when, for a given modal logic proof system, there exists another modal logic proof system such that proofs in the first are equivalent to cyclic proofs in the second. There, she shows that iGL, an[1] intuitionistic version of GL, also has a natural cyclic proof system.

We provide an alternative proof of the equivalence of a standard calculus for iGL and a cyclic one. The difference from the above-mentioned previous work is twofold:

1. Part of the motivation of [6] is not to use a non-wellfounded system as in [8]. We want to use a non-wellfounded system and in particular define the proof translation by corecursion.
2. In [8] and [6] cyclic systems are defined such that every cycle is allowed. We are going to see that for the sequent calculus of iGL that we work with this is not possible. The difference is that for propositional logic we will use the rules of the intuitionistic calculus G3i (see [9]). As we have already mentioned, Iemhoff's method can be applied to obtain a cyclic calculus for iGL, but using Dyckhoff calculus (see [4]) instead of G3i. See Section 3.1 for details.

2 Algebras and Coalgebras: (co)recursion[2]

We need to work with non-wellfounded trees. We are going to use the representation of trees with a non-empty set of finite sequences of natural numbers. Given a set A we will write A^* to denote the set of finite sequences over A. Elements of ω^* will be denoted by w, v, u.

Definition 1. An *L-labelled tree* is a pair $T = (N, \ell)$, where:

*This work is based on a master project done at the Master of Logic in the University of Amsterdam. I thank Bahareh Afshari and Lide Grotenhuis for supervising the project.

[1]The use of "an" instead of "the" is deliberate. Check the footnote of page 16 for an explanation.

[2]We are just going to introduce the necessary principles for our work, particularly infinite trees and corecursion to them. These ideas can already found in [2], so the methods we use can be considered to be standard for the treatment of infinite trees. For a general introduction on coalgebras the reader can consult [11].

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 13–20, doi:10.4204/EPTCS.435.2

1. $N \subseteq \omega^*$, non-empty and closed under initial segments.
2. For any $w \in N$ there exists a natural number m such that for any i, $wi \in N$ iff $i < m$. Given w, this number can be shown to be unique and we call it the *arity of w in T*, T-$\mathsf{arity}(w)$.
3. $\ell : N \longrightarrow L$, called the labelling function of T.

T is said to be *finite* iff N is finite. A branch of T is just an infinite path of T. We denote the collection of L-labelled trees as $\mathbb{T}_L$ and the collection of finite L-labelled trees as $\mathbb{T}_L^{<\omega}$. In case it can be filled by context, we will omit L.

Given a finite or infinite sequence w and a natural number i, we will write $w{\restriction}i$ to mean the restriction of w to $\{0,\ldots,i-1\}$. If X is a set of finite sequences, we say that w is maximal in X iff there is no sequence in X strictly extending w. Let us define some usual notions of trees in this formalism.

Definition 2. A *branch* of T is just an infinite sequence $b \in \omega^\omega$ such that for any $i \in \omega$, $b{\restriction}i$ is a node of T.

A *leaf* of T is a maximal sequence in the nodes of T. An *internal node* is any node that is not a leaf.

Let $T = (N, \ell)$ be a tree and w one of its nodes. We define the *T-subtree generated at w*, as $\mathsf{ST}_w(T) = (N', \ell')$ where:

$$N' = \{v \in \omega^* \mid wv \in N\},$$
$$\ell'(v) = \ell(wv).$$

We need to explain how to do corecursion over trees. In order to do so, we are going to use category theory and define recursion at the same time. First we need to define algebras, coalgebras and the morphisms between them.

Definition 3 (Algebra/Coalgebra)**.** Let F be an endofunctor of the category **Set**. An F-algebra is a pair (A, α) of a set A and a function $\alpha : F(A) \longrightarrow A$. An algebra morphism from (A, α) to (B, β) is just a function $f : A \longrightarrow B$ such that the following diagram commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{Ff} & F(B) \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A & \xrightarrow[f]{} & B \end{array}$$

Similarly, an F-coalgebra is a pair (C, γ) of a set C and a function $\gamma : C \longrightarrow F(C)$. A coalgebra morphism from (C, γ) to (D, δ) is a function $g : C \longrightarrow D$ such that the following diagram commutes:

$$\begin{array}{ccc} C & \xrightarrow{g} & D \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\delta} \\ F(C) & \xrightarrow[Fg]{} & F(D) \end{array}$$

An intuition to think about algebras and coalgebras is to imagine the functor F is representing some structure. Then, the objects of an algebra A are *created* using the structures of shape F over A. Conversely, the objects of a coalgebra C are *destructed* into structures of shape F over C. Using (co)algebras as objects and (co)algebra morphisms as arrows, we can define a category we will call it **F-(Co)Alg**. We say that a (co)algebra is initial (final), in case it is the[3] initial (final) object of the corresponding category. Finally, we can define what it means to define a function by recursion or corecursion.

[3] We are justified to talk about the initial (final) (co)algebra, since if it exists it is unique up to (co)algebra isomorphism.

Definition 4 (Recursion/Corecursion)**.** Let (A,α) be the initial algebra of an endofunctor F and B be a set. We say that $f : A \longrightarrow B$ has been defined by *recursion* iff there exists a function $\beta : F(B) \longrightarrow B$ such that f is the only algebra morphism from (A,α) to (B,β).

Let (D,δ) be the final coalgebra of an endofunctor F and C be a set. We say that $g : C \longrightarrow D$ has been defined by *corecursion* iff there exists a function $\gamma : C \longrightarrow F(C)$ such that g is the only coalgebra morphism from (C,γ) to (D,δ).

We can define an endofunctor of **Set**, $\mathscr{T}_L$, such that non-wellfounded (finitely branching) L-labelled trees are its final coalgebra and finite L-labelled trees are its initial algebra.

Definition 5 (Tree endofunctor)**.** Let L be a set of labels. We define the **Set** endofunctor $\mathscr{T}_L$ as:

$$\begin{aligned}\mathscr{T}_L(A) &= L \times A^*,\\ \mathscr{T}_L(f : A \longrightarrow B) &= \mathrm{id}_L \times \mathrm{map}_f,\end{aligned}$$

where $\mathrm{map}_f : A^* \longrightarrow B^*$ is the pointwise application of f.

We need to find the functions that make the finite trees an initial algebra and the non-wellfounded trees a final coalgebra. These functions are well-known, we call them $\mathsf{construct} : L \times \mathbb{T}^* \longrightarrow \mathbb{T}$ and $\mathsf{destruct} : \mathbb{T} \longrightarrow L \times \mathbb{T}^*$.

Definition 6. We define the function $\mathsf{construct} : L \times \mathbb{T}^* \longrightarrow \mathbb{T}$ as $\mathsf{construct}(a,((N_0,\ell_0),\ldots,(N_{n-1},\ell_{n-1}))) = (N,\ell)$ where:

$$N = \{\varepsilon\} \cup \bigcup_{i<n} \{iw \mid w \in N_i\},$$

$$\ell(w) = \begin{cases} a & \text{if } w = \varepsilon,\\ \ell_i(v) & \text{if } w = iv.\end{cases}$$

We define the function $\mathsf{destruct} : \mathbb{T} \longrightarrow L \times \mathbb{T}^*$ as:

$$\mathsf{destruct}(N,\ell) = (\ell(\varepsilon),(\mathsf{succ}_i(N,\ell))_{i<T\text{-}\mathsf{arity}(\varepsilon)})$$

where $\mathsf{succ}_i(T) = \mathsf{ST}_i(T)$ for $i < T\text{-}\mathsf{arity}(\varepsilon)$, called the i-th successor of T.

In simple words, given a label a and a finite sequence of trees $T_0,\ldots,T_{n-1}$ we have that $\mathsf{construct}$ creates the tree whose root has a as label and T_i as the i-th successor. Similarly given a tree T, $\mathsf{destruct}$ will return a pair with the label of the root and the sequence of trees which are successors of the root, in order. Note that if we restrict the domain to finite trees, we will obtain finite trees in the codomain for both functions. It is easy to check that:

Lemma 7. $(\mathbb{T},\mathsf{destruct})$ is the final coalgebra of $\mathscr{T}_L$ and $(\mathbb{T}^{<\omega},\mathsf{construct})$ is the initial algebra of $\mathscr{T}_L$.

In other words, we can define functions to trees by corecursion and from finite trees by recursion.

3 Non-wellfounded and cyclic proofs

We work with formulas in the language described by the following BNF:

$$\phi ::= p \mid \bot \mid \phi \to \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \Box\phi,$$

where p is a propositional variable. Note that, since our base logic is intuitionistic, $\Diamond$ is not definable with $\Box$. In other words, we are working in the $\Box$-fragment of modal logic.[4]

A sequent is just a pair (Γ, ϕ) where Γ is a finite multiset of formulas and ϕ is a formula. We will use Seq to denote the set of all sequents. In other words, we work with 2-sided single conclusion sequents. A *rule instance* is a pair consisting in a finite sequence of sequents and a sequence, called premises and conclusion. A *rule* is just a set of rule instances, let Rul be the set consisting in all rules (i.e. the set with all sets of rule instances). We are interested in the following rules:

$$\frac{}{\Gamma, p \Rightarrow p}\,\text{Prop} \qquad \frac{}{\Gamma, \bot \Rightarrow \phi}\,\text{Abs}$$

$$\frac{\Gamma, \phi, \psi \Rightarrow \chi}{\Gamma, \phi \wedge \psi \Rightarrow \chi}\,\wedge\text{L} \qquad \frac{\Gamma \Rightarrow \phi \quad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \wedge \psi}\,\wedge\text{R}$$

$$\frac{\Gamma, \phi \Rightarrow \chi \quad \Gamma, \psi \Rightarrow \chi}{\Gamma, \phi \vee \psi \Rightarrow \chi}\,\vee\text{L} \qquad \frac{\Gamma \Rightarrow \phi}{\Gamma \Rightarrow \phi \vee \psi}\,\vee\text{R}_1 \quad \frac{\Gamma \Rightarrow \psi}{\Gamma \Rightarrow \phi \vee \psi}\,\vee\text{R}_2$$

$$\frac{\Gamma, \phi \to \psi \Rightarrow \phi \quad \Gamma, \psi \Rightarrow \chi}{\Gamma, \phi \to \psi \Rightarrow \chi}\,\to\text{L} \qquad \frac{\Gamma, \phi \Rightarrow \psi}{\Gamma \Rightarrow \phi \to \psi}\,\to\text{R}$$

$$\frac{\Gamma, \Box\Gamma \Rightarrow \phi}{\Pi, \Box\Gamma \Rightarrow \Box\phi}\,\Box_{\mathsf{K4}} \qquad \frac{\Gamma, \Box\Gamma, \Box\phi \Rightarrow \phi}{\Pi, \Box\Gamma \Rightarrow \Box\phi}\,\Box_{\mathsf{GL}}$$

where Γ, Π are multisets of formulas, p is a propositional variable and ϕ, ψ, χ are formulas. The first 9 rules are called propositional rules, while the last 2 are called modal rules.

From now on, we assume that with tree we mean Seq $\times$ Rul-labelled tree. This permits us to talk about the premises, conclusion and rule of any node w as follows.

Definition 8. Let π be a tree and w one of its nodes. We define:

$$\begin{aligned}
&\pi\text{-prem}(w) = (\mathsf{fst}(\ell(wi)))_{i<\pi\text{-}\mathsf{arity}(w)},\\
&\pi\text{-concl}(w) = \mathsf{fst}(\ell(w)),\\
&\pi\text{-rule}(w) = \mathsf{snd}(\ell(w)),
\end{aligned}$$

where fst is the first projection from an ordered pair and snd the second projection.

A proof in iGL is just a standard finite prooftree generated by the propositional logic rules and the modal rule $\Box_{\mathsf{GL}}$. Let us define non-wellfounded and cyclic proofs in iK4.

Definition 9. A *non-wellfounded proof in* iK4 is a tree π such that:

1. For any node w, $(\pi\text{-prem}(w), \pi\text{-concl}(w))$ is an instance of the rule $\pi\text{-rule}(w)$ and $\pi\text{-rule}(w)$ is either a propositional rule or $\Box_{\mathsf{K4}}$.
2. (Progress condition) For any branch w, there are infinitely many i's with $\pi\text{-rule}(w\restriction i) = \Box_{\mathsf{K4}}$.

We will write $\vdash_{\mathsf{iK4}_\infty} S$ to mean that π is a non-wellfounded proof in iK4 and $\pi\text{-concl}(\varepsilon) = S$. Also, we will denote the collection of non-wellfounded proofs in iK4 as $\mathbb{P}(\mathsf{iK4}_\infty)$.

[4] In particular iGL will be the smallest modal logic with intuitionistic propositional logic and the non-logical axioms of GL formulated with $\Box$. There is an alternative approach to intuitionistic GL to consider also the diamond intuitionistically, called IGL. Note that even the $\Box$-fragment of these logics is not the same, so they must not be identified. The reader interested in IGL should consult [3].

Definition 10. A *cyclic proof in* iK4 is a pair (τ, b) such that:

1. τ is a finite tree and b is a partial function from leaves of τ to the internal nodes of τ called the *backlink function*.
2. For any node $w \notin \mathrm{dom}(b)$, we have that $(\pi\text{-prem}(w), \pi\text{-concl}(w))$ is an instance of the rule $\pi\text{-rule}(w)$ and $\pi\text{-rule}(w)$ is a propositional rule or $\Box_{\mathsf{K4}}$.
3. For any node $w \in \mathrm{dom}(b)$, we have that:
   (a) $b(w)$ is a (strict) initial segment of w.
   (b) $\pi\text{-concl}(w) = \pi\text{-concl}(b(w))$ and $\pi\text{-rule}(w) = \varnothing$.
   (c) (Progress condition) There is a v between $b(w)$ and w ($b(w)$ initial segment of v and v initial segment of w), with $\pi\text{-rule}(v) = \Box_{\mathsf{K4}}$.

We will write $\vdash_{\mathsf{iK4}_\circ} S$ to mean that π is a cyclic proof in iK4 and $\pi\text{-concl}(\varepsilon) = S$. Also, we will denote the collection of cyclic proofs in iK4 as $\mathbb{P}(\mathsf{iK4}_\circ)$.

3.1 On the need of progress

Let us show that with the sequent rules we have chosen we explicitly need a progress condition. For that simply consider the following cyclic proof:

$$\dfrac{\dfrac{}{p \to q, q \to p \Rightarrow p}\,\text{Cycle to (i)} \qquad \dfrac{\dfrac{}{q, q \to p \Rightarrow q}\,\text{Prop} \quad \dfrac{}{q, p \Rightarrow p}\,\text{Prop}}{q, q \to p \Rightarrow p}\,{\to}\text{R}}{\text{(i)} \quad p \to q, q \to p \Rightarrow p}\,{\to}\text{R}$$

Clearly $p \to q, q \to p \Rightarrow p$ should not be a provable sequent. Since this cyclic proof can also be seen as an infinitary proof it follows that the progress condition is needed in both systems.

4 Infinitary Proof Translation

Thanks to the definition of corecursion we will be able to define a proof translation from a function $\alpha : \mathbb{T}^{<\omega} \longrightarrow (\mathrm{Seq} \times \mathrm{Rul}) \times (\mathbb{T}^{<\omega})^*$. However, not any function of that shape will give a function from proofs to proofs, in the following definition we enumerate the necessary conditions for this to happen.

Definition 11 (Infinitary Proof Translation)**.** Let $\alpha : \mathbb{T}^{<\omega} \longrightarrow (\mathrm{Seq} \times \mathrm{Rul}) \times (\mathbb{T}^{<\omega})^*$. We say that it is an infinitary proof translation iff for any $\pi \in \mathbb{P}(\mathsf{iGL})$, if we denote:

$$\alpha(\pi) = ((S, R), (\tau_0, \ldots, \tau_{n-1})),$$
$$\alpha(\tau_i) = ((S_i, R_i), \ldots),$$

then the following conditions are satisfied:

1. $\tau_0, \ldots, \tau_{n-1} \in \mathbb{P}(\mathsf{iGL})$.
2. The following is a rule instance of iK4:

$$\dfrac{S_0 \quad \cdots \quad S_{n-1}}{S}\,R$$

3. If $R \neq \Box_{\mathsf{K4}}$, then $\text{height}(\tau_0), \ldots, \text{height}(\tau_{n-1}) < \text{height}(\pi)$.

Given such α we define trans_α as the only coalgebra morphism from $(\mathbb{T}^{<\omega}, \alpha)$ to $(\mathbb{T}, \mathsf{destruct})$. This implies that

$$\text{trans}_\alpha = \mathsf{construct} \circ (\text{id} \times \mathsf{map}_{\text{trans}_\alpha}) \circ \alpha.$$

We want to show that if α is an infinitary proof translation and π is a (finite) proof in iGL, then $\text{trans}_\beta(\pi)$ is a non-wellfounded proof in iK4. First we need the following technical lemma:

Lemma 12. Let α be an infinitary proof translation and $\pi \in \mathbb{P}(\mathsf{iGL})$. If w is a node of $\text{trans}_\alpha(\pi)$, then there is an unique sequence of finite iGL-proofs, $(\iota_i)_{i \leq \text{length}(w)}$, such that $\iota_0 = \pi$, $\mathsf{ST}_{w\restriction i}(\text{trans}_\alpha(\pi)) = \text{trans}_\alpha(\iota_i)$ for $i \leq \text{length}(w)$ and $\iota_{i+1} = \mathsf{succ}_{w_i}(\alpha(\iota_i))$ for $i < \text{length}(w)$.

Similarly, if w is a branch of $\text{trans}_\alpha(\pi)$, then there is an unique sequence of finite iGL-proofs, $(\iota_i)_{i \in \omega}$, such that $\iota_0 = \pi$, $\mathsf{ST}_{w\restriction i}(\text{trans}_\alpha(\pi)) = \text{trans}_\alpha(\iota_i)$ and $\iota_{i+1} = \mathsf{succ}_{w_i}(\alpha(\iota_i))$ for $i \in \mathbb{N}$.

Proof. The result for nodes is proven by induction in the length of the node. To prove the result for a branch b it is enough take the union of the sequences given by applying the result for nodes to $b\restriction i$ for each $i \in \mathbb{N}$. □

With this lemma we can show that infinitary proof translations transform finite proofs in iGL into infinitary proofs in iK4:

Theorem 13. If α is an infinitary proof translation, then

$$\text{trans}_\alpha : \mathbb{P}(\mathsf{iGL}) \longrightarrow \mathbb{P}(\mathsf{iK4}_\infty).$$

Proof. Let $\iota \in \mathbb{P}(\mathsf{iGL})$, we have to check that every node in $\pi = \text{trans}_\alpha(\iota)$ is an instance of a rule of iK4 and that the progress condition is fulfilled.

Proof that every node is the instance of a rule. Let w be a node of π of length n. By Lemma 12 we have that there is a sequence of finite trees $(\iota_i)_{i \leq n}$. By the statement of the Lemma $\iota_0 = \iota$ and $\iota_{i+1} = \mathsf{succ}_{w_i}(\alpha(\iota_i))$. This together with the first condition of proof translation gives that for each i, $\iota_i \in \mathbb{P}(\mathsf{iGL})$. Using Lemma 12 with wj for $j < \pi\text{-arity}(w)$ we get sequences $(\iota_i^j)_{i \leq n+1}$ such that $\iota_i^j = \iota_i$ for $i \leq n$ and $\iota_{n+1}^j = \mathsf{succ}_j(\alpha(\iota_n))$. We note that since $\mathsf{ST}_w(\text{trans}_\alpha(\iota)) = \text{trans}_\alpha \iota_n$ and $\mathsf{ST}_{wj}(\text{trans}_\alpha(\iota)) = \text{trans}_\alpha \iota_{n+1}^j$ the conclusion of rule instance at w in π is just the result of looking at the sequent given by applying α tp ι_n and the premises to is the sequent given by applying α to each ι_n^j. Using this and the second condition of infinitary proof translation to ι_n, since $\iota_n \in \mathbb{P}(\mathsf{iGL})$. we get that the node is the instance of a rule, as desired.

Proof of progress condition. Imagine the result does not fulfill the progress condition, then there is an infinite branch b in π such that from some point do not contain applications of $\Box_{\mathsf{K4}}$. By applying Lemma 12 to this branch b we would obtain an infinite sequences of finite trees $(\iota_n)_{n \in \mathbb{N}}$. By the third condition of infinitary proof translation we would obtain that from some point the height of these trees is strictly decreasing, absurd. □

5 From finitary to cyclic, and vice versa

In order to define the translation from the finite acyclic system into the cyclic system first we need to obtain non-wellfounded proofs of certain shape. For this, we need the admissibility of the following two rules in iGL:

$$\frac{\Gamma, \psi, \psi \Rightarrow \phi}{\Gamma, \psi \Rightarrow \phi}\,\text{Contract} \qquad \frac{\Gamma, \Box\Gamma, \Box\phi \Rightarrow \phi}{\Gamma, \Box\Gamma \Rightarrow \phi}\,\text{Löb}$$

The admissibility of these rules is proven in [5]. If Γ is a finite multiset it can be split into a set (a multiset where all its elements appear exactly once) and a multiset in a unique way. Γ^s will be the resulting set of this splitting and Γ^m will be the multiset.

Thanks to admissibility of contraction we can define a function that, given an iGL-proof π of $\Gamma, \Box\Gamma \Rightarrow \phi$, returns a proof $\mathsf{contract}(\pi)$ of $\Gamma^s, \Box\Gamma^s \Rightarrow \phi$. Similarly, thanks to admissibility of Löb we can define a function that given an iGL-proof π of $\Gamma, \Box\Gamma, \Box\phi \Rightarrow \phi$, returns a proof $\mathsf{löb}(\pi)$ of $\Gamma, \Box\Gamma \Rightarrow \phi$.

Using these two functions and the theorem of proof translations, we have the following result:

Theorem 14. There is a unique $h : \mathbb{P}(\mathsf{iGL}) \longrightarrow \mathbb{P}(\mathsf{iK4}_\infty)$, such that h is the identity in initial sequents, commutes with the logical rules and:

$$\frac{\pi}{\dfrac{\Gamma, \Box\Gamma, \Box\phi \Rightarrow \phi}{\Pi, \Box\Gamma \Rightarrow \Box\phi}\,\Box_{\mathsf{GL}}} \longmapsto \frac{h(\mathsf{contract}(\mathsf{löb}(\pi)))}{\dfrac{\Gamma^s, \Box\Gamma^s \Rightarrow \phi}{\Pi, \Box\Gamma^m, \Box\Gamma^s \Rightarrow \Box\phi}\,\Box_{\mathsf{K4}}}$$

In addition, if π is a proof of $\Gamma \Rightarrow \phi$, then $h(\pi)$ is also a proof of $\Gamma \Rightarrow \phi$.

Proof. Simple application of 13. □

Finally, thanks to this proof translation we can show that any finite acyclic proof can be transformed into a cyclic proof.

Corollary 15. Let S be a sequent. If $\vdash_{\mathsf{iGL}} S$, then $\vdash_{\mathsf{iK4}_\circ} S$.

Proof. This result is based in two observations. First, that thanks to the shape of the rules we have the subformula property. Second, that any branch must have infinitely many applications of the $\Box_{\mathsf{K4}}$ rule, but choosing the translation we defined before the premise of such an application is determined by a set. Using these two facts together we can get the required repetition of sequents to define a cyclic proof. □

The argument to obtain a proof from the cyclic system to the finite acyclic system is totally analogous to the classical case proven in [8]. We can conclude the desired result:

Theorem 16. Let S be a sequent. Then $\vdash_{\mathsf{iGL}} S$ iff $\vdash_{\mathsf{iK4}^\circ} S$.

6 Conclusion

We have provided a proof of the equivalence between a finite acyclic system with rules of iGL and a (finite) cyclic system with rules of iK4 using a corecursive translation of proofs. In order to do this, we exploited that while performing a corecursion we can use the admissible rules in the finite acyclic system to obtain non-wellfounded proofs of the desired shape.

References

[1] James Brotherston (2006): *Sequent calculus proof systems for inductive definitions*. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics. Available at `http://hdl.handle.net/1842/1458`.

[2] Bruno Courcelle (1983): *Fundamental properties of infinite trees*. *Theoretical Computer Science* 25(2), pp. 95–169, doi:10.1016/0304-3975(83)90059-2.

[3] Anupam Das, Iris van der Giessen & Sonia Marin (2024): *Intuitionistic Gödel-Löb Logic, à la Simpson: Labelled Systems and Birelational Semantics*. In Aniello Murano & Alexandra Silva, editors: *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 288, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 22:1–22:18, doi:10.4230/LIPIcs.CSL.2024.22.

[4] Roy Dyckhoff (1992): *Contraction-Free Sequent Calculi for Intuitionistic Logic*. *The Journal of Symbolic Logic* 57(3), pp. 795–807, doi:10.2307/2275431.

[5] Iris van der Giessen & Rosalie Iemhoff (2021): *Sequent Calculi for Intuitionistic Gödel–Löb Logic*. *Notre Dame Journal of Formal Logic* 62(2), pp. 221 – 246, doi:10.1215/00294527-2021-0011.

[6] Rosalie Iemhoff (2016): *Reasoning in circles*. *Liber Amicorum Alberti: A Tribute to Albert Visser*.

[7] Yury Savateev & Daniyar Shamkanov (2018): *Non-Well-Founded Proofs for the Grzegorczyk Modal Logic*. *The Review of Symbolic Logic* 14, doi:10.1017/S1755020319000510.

[8] D. S. Shamkanov (2014): *Circular proofs for the Gödel-Löb provability logic*. *Mathematical Notes* 96(3–4), p. 575–585, doi:10.1134/s0001434614090326.

[9] A. S. Troelstra & H. Schwichtenberg (2000): *Basic Proof Theory*, 2 edition. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, doi:10.1017/CBO9781139168717.

[10] Guillermo Menéndez Turata (2024): *Cyclic Proof Systems for Modal Fixpoint Logics*. Ph.D. thesis, University Of Amsterdam. Faculty of Science. Institute for Logic, Language and Computation (ILLC). Available at `https://eprints.illc.uva.nl/id/eprint/2285`.

[11] Yde Venema (2007): *Algebras and coalgebras*. In Patrick Blackburn, Johan Van Benthem & Frank Wolter, editors: *Handbook of Modal Logic*, *Studies in Logic and Practical Reasoning* 3, Elsevier, pp. 331–426, doi:10.1016/S1570-2464(07)80009-7.

Cut elimination for Cyclic Proofs: A Case Study in Temporal Logic*

Bahareh Afshari
University of Gothenburg
Gothenburg, Sweden
bahareh.afshari@gu.se

Johannes Kloibhofer
University of Amsterdam
Amsterdam, The Netherlands
j.kloibhofer@uva.nl

We consider modal logic extended with the well-known temporal operator 'eventually' and provide a cut-elimination procedure for a cyclic sequent calculus that captures this fragment. The work showcases an adaptation of the reductive cut-elimination method to cyclic calculi. Notably, the proposed algorithm applies to a cyclic proof and directly outputs a cyclic cut-free proof without appealing to intermediate machinery for regularising the end proof.

1 Introduction

Non-wellfounded derivation systems underwent their first rigorous study in [9, 11, 2, 14]. In contrast to the traditional finite derivation trees, one allows for infinite branches that mimic the induction axioms required for capturing semantics of temporal operators (more generally, implicit/explicit fixpoints). *Cyclic proofs* constitute the regular fragment of non-wellfounded proofs and are closer in spirit, and often in practice, to finitary proof systems with induction axioms. To exclude vicious reasoning and capture the true semantics of temporal constructs, a so called *global validity condition* is placed on infinite branches. This condition necessitates a re-investigation of reductive cut elimination in the non-wellfounded setting.

With the advance of non-wellfounded and cyclic proof theory, cut elimination has also received considerable attention (see e.g. [5, 3, 13, 12]). The common approach proceeds in two steps: first cuts are pushed 'up' away from the root by the usual cut reductions to obtain, in the limit, a non-wellfounded derivation; in the second step the acquired limit structure is shown to satisfy the global validity condition. When it comes to cyclic systems, cut-elimination procedures that can *directly* produce cut-free cyclic proofs are rare. Although the aforementioned two-step approach can be applied, the resulting structure may not necessarily be a regular tree. For those calculi whose cyclic fragment does exhaust all validities one may invoke other machinery, such as automata, to find a cut-free cyclic proof.

We consider a simple temporal logic MLe with a complete cyclic sequent calculus GKe and describe a syntactic cut-elimination procedure for GKe that works directly on cyclic proofs. The logic MLe extends modal logic with the so called 'eventually' operator, the reflexive and transitive closure of the diamond modality. Although MLe is expressively weak, this work can be seen as a starting point to apply cut elimination to cyclic proof systems for richer modal fixpoint logics.

Cuts occurring in a cyclic proof can be split into two categories: cuts that reside in a cycle and those that do not. We call them, respectively, *unimportant* and *important* cuts. We show that unimportant cuts are for most part harmless, in the sense that the cut formula does not interfere with the global validity condition and as such, these cuts can be eliminated as usual by pushing them away from the root of the

*This work was supported by the Dutch Research Council [OCENW.M20.048, 617.001.857] and Knut and Alice Wallenberg Foundation [2020.0199].

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 21–40, doi:10.4204/EPTCS.435.3

proof-tree. The main challenge posed is to eliminate important cuts, as in this case the cut formula may contribute to the global validity condition.

The global validity condition can take different forms. A *trace-based condition* is formulated in terms of traces, i.e. paths of ancestry of formulas, on a branch. Such conditions are relatively easy to formulate but harder to work with. By annotating sequents with additional information one may succeed in formulating a *branch-based condition* that is not directly concerned with traces. Such a condition is easier to work with when performing operations such as projecting a proof or zipping two proofs that are critical in reductive cut elimination. Annotated proof systems for modal μ-calculus [1] were first introduced by Jungteerapanich and Stirling [6, 15] building on earlier work by Niwiński and Walukiewicz [9]. There are general ways of obtaining annotated proof systems from cyclic proof systems [4, 7]. It is with this rich framework in mind that the presented case analysis has been carried out.

2 Modal Logic with Eventuality

Let Prop be an infinite set of propositions. *Formulas* of MLe are defined inductively by

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \mathsf{F}\varphi$$

where $p \in \mathsf{Prop}$. The connectives $\vee, \rightarrow$ and $\Box$ are taken as abbreviations and can be defined as usual. To see MLe as a fragment of the modal μ-calculus, simply define $\mathsf{F}\varphi \equiv \mu x.(\varphi \vee \Diamond x)$.

Semantics of MLe-formulas is given with respect to Kripke models in the standard way. A *Kripke model* is a tuple $\mathbb{S} = (S, R, V)$, where S is a non-empty set, R is a binary relation on S and V is a function $S \rightarrow \mathscr{P}(\mathsf{Prop})$. Let $s \in S$ and define R^* to be the reflexive and transitive closure of R. The relation $\Vdash$ is defined inductively by:

$\mathbb{S}, s \Vdash p$	$\Leftrightarrow$	$p \in V(s)$
$\mathbb{S}, s \Vdash \neg\varphi$	$\Leftrightarrow$	$\mathbb{S}, s \nVdash \varphi$
$\mathbb{S}, s \Vdash \varphi \wedge \psi$	$\Leftrightarrow$	$\mathbb{S}, s \Vdash \varphi$ and $\mathbb{S}, s \Vdash \psi$
$\mathbb{S}, s \Vdash \Diamond\varphi$	$\Leftrightarrow$	there exists $t \in S$ with sRt and $\mathbb{S}, t \Vdash \varphi$
$\mathbb{S}, s \Vdash \mathsf{F}\varphi$	$\Leftrightarrow$	there exists $t \in S$ with sR^*t and $\mathbb{S}, t \Vdash \varphi$

A formula φ is called *valid* if $\mathbb{S}, s \Vdash \varphi$ for every Kripke model $\mathbb{S} = (S, R, V)$ and every $s \in S$.

We present the cyclic annotated proof system GKe, which is sound and complete for MLe. The annotations allow us to formulate the global validity condition as a simple condition on paths. Since MLe is a fragment of the alternation-free μ-calculus, as established in [8], one annotation mark per formula suffices for the formulation of the validity condition. Moreover, MLe-traces are very well-behaved; in particular, cyclic traces do not pass through disjunctions, requiring at most one annotated formula in each sequent, as demonstrated in [10].

An *annotated formula* is a pair (φ, a), usually denoted as φ^a, where $a \in \{f, u\}$. We call annotated formulas of the form φ^f *in focus* and φ^u *out of focus*. Let Γ and Δ be *sets* of annotated formulas, where every formula in Δ is out of focus and at most one formula in Γ is in focus and this formula has the form $\mathsf{F}\varphi$ or $\Diamond\mathsf{F}\varphi$ for some φ. Then $\Gamma \Rightarrow \Delta$ is called an *annotated sequent*. As every formula in Δ is out of focus, we omit the annotations in the right side of the annotated sequent. We define $\Diamond\Delta = \{\Diamond\varphi \mid \varphi \in \Delta\}$ and, for a set of annotated formulas Γ, we set $\Gamma^u = \{\varphi^u \mid \varphi^a \in \Gamma\}$. If it is clear from the context, we will call annotated formulas/sequents simply formulas/sequents.

$$\mathsf{Ax}\colon \frac{}{p^u \Rightarrow p} \qquad \Diamond\colon \frac{\varphi^a \Rightarrow \Delta}{\Diamond\varphi^a \Rightarrow \Diamond\Delta} \qquad \wedge_L\colon \frac{\varphi^u, \psi^u, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi^u, \Gamma \Rightarrow \Delta} \qquad \wedge_R\colon \frac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi}$$

$$\mathsf{F}_L\colon \frac{\Diamond\mathsf{F}\varphi^a, \Gamma \Rightarrow \Delta \quad \varphi^u, \Gamma \Rightarrow \Delta}{\mathsf{F}\varphi^a, \Gamma \Rightarrow \Delta} \qquad \mathsf{F}_R\colon \frac{\Gamma \Rightarrow \Delta, \varphi, \Diamond\mathsf{F}\varphi}{\Gamma \Rightarrow \Delta, \mathsf{F}\varphi} \qquad \neg_L\colon \frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi^u, \Gamma \Rightarrow \Delta} \qquad \neg_R\colon \frac{\varphi^u, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi}$$

$$\mathsf{cut}\colon \frac{\Gamma_l \Rightarrow \Delta_l, \varphi \quad \varphi^u, \Gamma_r \Rightarrow \Delta_r}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r} \qquad \mathsf{w}_L\colon \frac{\Gamma \Rightarrow \Delta}{\varphi^u, \Gamma \Rightarrow \Delta} \qquad \mathsf{w}_R\colon \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

$$\mathsf{D}^{\mathsf{x}}\colon \frac{\begin{array}{c}[\Gamma \Rightarrow \Delta]^{\mathsf{x}} \\ \vdots \end{array}}{\Gamma \Rightarrow \Delta} \qquad \mathsf{u}\colon \frac{\varphi^f, \Gamma \Rightarrow \Delta}{\varphi^u, \Gamma \Rightarrow \Delta} \qquad \mathsf{f}\colon \frac{\varphi^u, \Gamma \Rightarrow \Delta}{\varphi^f, \Gamma \Rightarrow \Delta}$$

Figure 1: Rules of GKe

The rules of GKe for the propositional connectives and the modal operator are standard. The rules F_L and F_R for the eventually operator F stem from the identity $\mathsf{F}\varphi \equiv \varphi \vee \Diamond\mathsf{F}\varphi$. In F_L the formula $\mathsf{F}\varphi^a$ is called *principal*. The rule F_L and the focus rules u and f are the only ones that change the annotations of formulas. The *discharge rule* D allows us to discharge leaves, that are labelled by sequents already occurring at one of its ancestors. To qualify as a GKe proof, this is only allowed if a certain success-condition is met. Each instance of D^{x} is labelled by a unique *discharge token* x taken from a fixed infinite set $\mathscr{D} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}, ...\}$.

We will read proof trees 'upwards', so nodes labelled by premises are viewed as children of nodes labelled by the conclusion of a rule. A node v is called an *ancestor* of a node u if there are nodes $v = v_0$, $\ldots$, $v_n = u$ with v_{i+1} being a child of v_i for $i = 0, ..., n-1$ and $n > 0$.

Definition 1 (Derivation)**.** A GKe *derivation* $\pi = (T, P, \mathsf{S}, \mathsf{R})$ is a quadruple such that (T, P) is a, possibly infinite, tree with nodes T and parent relation $P \subseteq T \times T$; S is a function that maps every node $u \in T$ to an annotated sequent; R is a function that maps every node $u \in T$ to either (i) the name of a rule in Figure 1 or (ii) a discharge token, such that (i) the specifications of the rules in Figure 1 are satisfied, (ii) every node labelled with a discharge token is a leaf, and (iii) for every leaf l that is labelled with a discharge token $\mathsf{x} \in \mathscr{D}$ there is an ancestor $c(l)$ of l that is labelled with D^{x} (and such that l and $c(l)$ are labelled with the same sequent). In the latter condition, we call l a *repeat leaf*, and $c(l)$ *companion* node of l.

A GKe derivation of a sequent $\Gamma \Rightarrow \Delta$ is a GKe derivation, where the root is labelled by $\Gamma \Rightarrow \Delta$.

Let $\pi = (T, P, \mathsf{S}, \mathsf{R})$ be a derivation. We will be working with the following two trees associated to π.

(i) The usual proof tree $\mathscr{T}_\pi = (T, P)$.

(ii) The *proof tree with back edges* $\mathscr{T}_\pi^C = (T, P^C)$ where P^C is the parent relation plus back-edges for each repeat leaf, i.e., $P^C = P \cup \{(l, c(l)) \mid l \text{ is a repeat leaf}\}$.

A *path* in a GKe derivation $\pi = (T, P, \mathsf{S}, \mathsf{R})$ is a path in $\mathscr{T}_\pi^C$.

Definition 2 (Successful path)**.** A path τ in a GKe derivation is called *successful* if

1. every sequent on τ has a formula in focus, and
2. τ passes through an application of F_L, where the principal formula is in focus.

Let v be a a repeat leaf in a GKe derivation $\pi = (T, P, \mathsf{S}, \mathsf{R})$ with companion $c(v)$, and let τ_v denote the path in (T, P) from $c(v)$ to v. We call v a *discharged leaf* if the path τ_v is successful. A leaf is called *closed* if it is either a discharged leaf or labelled by Ax and *open*, otherwise.

Definition 3 (Proof)**.** A GKe *proof* π is a finite GKe derivation, where every leaf is closed.

The next lemma is usually a consequence of guardedness. For the GKe proof system this is immediate.

Lemma 4. *If v is a discharged leaf in a* GKe *proof, then there is a node labelled by $\Diamond$ on τ_v.*

The soundness and completeness of GKe follows from [10] wherein a cyclic hypersequent calculi is given for a class of modal logics with the master modality characterised by frame conditions. In the case of no frame conditions, the calculus of [10], which employs the dual modalities $\neg\Diamond\neg\varphi$ and $\neg\mathsf{F}\neg\varphi$ as primitive, coincides with GKe.

Theorem 5 (Soundness and Completeness)**.** *There is a* GKe *proof of $\Gamma \Rightarrow \Delta$ iff $\bigwedge\Gamma \to \bigvee\Delta$ is valid.*

3 Cut Elimination

We introduce a cut-elimination method based on reductive cut elimination but tailored to cyclic proofs. The cut reductions are the usual ones save for the additional care that must be taken when a discharge rule is duplicated. The reductions are listed in Appendix A.

The main difficulty in this approach are cut reductions, that alter successful paths. As an example consider the principal cut

$$\mathsf{cut}\ \dfrac{\mathsf{F}_R\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi, \Diamond\mathsf{F}\varphi}}{\Gamma \Rightarrow \Delta, \mathsf{F}\varphi} \qquad \mathsf{u}\ \dfrac{\mathsf{F}_L\ \dfrac{\overset{\pi_1}{\Diamond\mathsf{F}\varphi^f, \Gamma \Rightarrow \Delta} \qquad \overset{\pi_2}{\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{F}\varphi^f, \Gamma \Rightarrow \Delta}}{\mathsf{F}\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

and its reduction to

$$\mathsf{cut}\ \dfrac{\mathsf{cut}\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi, \Diamond\mathsf{F}\varphi} \qquad \mathsf{u}\ \dfrac{\overset{\pi_1}{\Diamond\mathsf{F}\varphi^f, \Gamma \Rightarrow \Delta}}{\Diamond\mathsf{F}\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_2}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

Note that in this reduction an F_L rule, where the principal formula is in focus, gets removed. Hence, paths that were successful before the reduction, could be unsuccessful afterwards. We define 'unimportant' cuts in such a way that in all cut reductions, successful paths remain successful, i.e. the behaviour from above may not occur. Thus, for unimportant cuts it suffices to push all cuts upwards until a repeat is reached. The treatment of important cuts is more complicated, as descendants of the cut-formulas could be in focus, and therefore being critical in the discharge condition of repeats. In order to make these observations formal we first need some technical definitions.

3.1 Preliminary definitions

Let (G, E) be a graph. A *strongly connected subgraph* of G is a set of nodes $A \subseteq G$, such that from every node of A there is a path to every other node in A. A *cluster* of G is a maximally strongly connected subgraph of G. A cluster is called *trivial* if it consists of only one node.

Let π be a GKe proof. A *cluster* of π is a cluster of $\mathscr{T}_\pi^C$. Let $\mathscr{S}_\pi$ be the set of non-trivial clusters of π. We define a relation $\twoheadrightarrow$ on $\mathscr{S}_\pi$ as follows: $S_1 \twoheadrightarrow S_2$ if $S_1 \neq S_2$ and there are nodes $v_1 \in S_1, v_2 \in S_2$ such

that there is a path from v_1 to v_2 in $\mathscr{T}_\pi^C$. The relation $\twoheadrightarrow$ is irreflexive, transitive and antisymmetric. We write $\mathsf{depth}(S)$ for the length of the longest path in $(\mathscr{S}_\pi, \twoheadrightarrow)$ starting from cluster S.

For a node v in a proof π, we define the *depth* of v to be

$$\mathsf{depth}(v) = \max\{\mathsf{depth}(S) \mid S \in \mathscr{S}_\pi \text{ and there is a path from } v \text{ to some } u \in S\}$$

where $\max \emptyset = 0$.

The *component* of v, written $\mathsf{comp}(v)$ is the set of nodes $u \in \pi$, that are reachable from v in $\mathscr{T}_\pi^C$ with $\mathsf{depth}(u) = \mathsf{depth}(v)$. Note that $\mathsf{comp}(v)$ does not have to coincide with a cluster in π, but may contain multiple clusters. The component of the root is called the *root-component* and the cluster of the root is called the *root-cluster*.

Any node v in a non-trivial cluster of a GKe proof π has a formula in focus, as it is on the path τ_l of a discharged leaf l to its companion. For nodes in a trivial cluster this is not necessarily the case. We can apply f and u rules in a certain way to minimise nodes with a formula in focus. By doing so, nodes with a formula in focus resemble the non-trivial clusters of the proof tree with back edges: Any node with a formula in focus is either in a non-trivial cluster or is labelled by f.

Formally, we call a GKe proof *minimally focused* if (i) if v is labelled by u, then its child is labelled by D and (ii) if $\mathsf{depth}(v) < \mathsf{depth}(v')$ for a child v of v', then v is labelled by f. As every proof can be transformed to a minimally focused proof of the same sequent where only u and f rules are added and removed, we will always assume that GKe proofs are minimally focused.

Definition 6 (Important cut)**.** Let C be an occurrence of a cut-rule in a GKe proof π. We call C *important*, if all formulas in the conclusion of C are out of focus; and *unimportant*, otherwise.

Definition 7 (Cut rank)**.** The *rank of a formula* φ, denoted $\mathsf{rank}(\varphi)$, is the maximal nesting depth of F's in φ.[1] The *rank of a* cut*-rule* is the rank of its cut formula. The *cut-rank of a proof* is the maximal rank amongst its cut-rules.

Let v' be a child node of v in a GKe derivation π. We call a formula φ' at v' an *immediate descendant* of φ at v if either (i) φ and φ' are designated formulas in the rule description or (ii) $\varphi = \varphi'$ and both are at the same side of the sequent. A formula ψ at a node u is called a *descendant* of a formula φ at a node v if there is a path $v = v_0, \ldots, v_n = u$ containing formulas $\varphi = \varphi_0, \ldots, \varphi_n = \psi$ respectively such that φ_{i+1} is an immediate descendant of φ_i for $i = 0, ..., n-1$. A descendant ψ at a node u of a formula φ at a node v is called a *component descendant* of φ if $u \in \mathsf{comp}(v)$.

The next lemma justifies the definition of unimportant cuts. It implies, that pushing unimportant cuts upwards does not alter successful paths. Note that this uses our assumption that GKe proofs are minimally focused.

Lemma 8. *Let* C *be an unimportant cut in a* GKe *proof* π*. Every component descendant of the cut-formula of* C *is out of focus.*

Definition 9 (Subproof)**.** For a node v in π, the *subproof of* π *rooted at* v, denoted π_v, is the result of recursively replacing every open leaf l in π_v with $\pi_{c(l)}$. In order to guarantee that D rules are labelled by unique discharge tokens, discharge tokens x are replaced by fresh discharge tokens, whenever a D^{x} rule is duplicated during a reduction step.

Note that π_v is well-defined, as v is a descendent of $c(l)$ for every open leaf l. Hence, at some point $\pi_{c(l)}$ has no open leaves.

[1] For example, the rank of $\mathsf{F}(\Diamond \mathsf{F} p \wedge \mathsf{F} \neg \Diamond \mathsf{F} p)$ is 3.

3.2 Important cut

The strategy to eliminate an important cut of rank n is as follows. We first push the cut upwards, until we reach a 'critical' cut. Then we take π_0 and π_1, the proofs of the left and right premise of this cut, and '*zip*' them together, while deleting all descendants of the cut formula. This is done recursively, similarly to how a cut can be pushed 'up'. In this process we treat π_0 and π_1 differently: In π_1 the formula in focus might be the cut formula and thus may be deleted, in π_0 we keep the same formulas in focus and successful repeats in π_0 will be projected to find successful repeats in our target proof.

In order to find suitable repeats we also have to keep track of the already constructed subproof. The intermediate objects of this process are called *traversed proofs*, that correspond to proofs, where on every branch of the proof there is at most one cut of rank n.

Definition 10 (Critical cut)**.** Let π be a GKe proof and C an important cut in π. We call C *critical* if the cut formula of C is of the form $\mathsf{F}\varphi$.

Lemma 11. *Let π be a GKe proof of cut-rank n where the only cut of rank n is important and at the root. Then there is a GKe proof π' of the same sequent with cut-rank n, where all cuts of rank n are critical.*

Proof. We can apply cut reductions from Appendix A until all cuts of rank n are critical. In all cut reductions, except where $\mathsf{F}\varphi$ is principal, the syntactic size (i.e., the number of symbols) of the cut formula is not increased, and in the reduction for $\Diamond$ the syntactic size of the cut formula decreases. Due to Lemma 4 there is a $\Diamond$ rule on the path from a companion node to all of its discharged leaves. Hence, the syntactic size of the cut formula ψ decreases until ψ is of the form $\mathsf{F}\varphi$. □

Definition 12 (Traversed proof)**.** An $\mathsf{F}\varphi$*-traversed proof* ρ of $\Sigma \Rightarrow \Pi$ is a derivation of $\Sigma \Rightarrow \Pi$, where all leaves are either closed or labelled by a sequent $\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r$ together with a triple (π_l, ψ, π_r), written as $[\pi_l]\psi[\pi_r]$, such that $\psi \equiv \mathsf{F}\varphi$ or $\psi \equiv \Diamond\mathsf{F}\varphi$, π_l is a GKe proof of $\Gamma_l \Rightarrow \Delta_l, \psi$ and π_r is a GKe proof of $\psi^a, \Gamma'_r \Rightarrow \Delta_r$, where $(\Gamma'_r)^u = \Gamma_r$. If $\mathsf{F}\varphi$ is clear from the context we will just write *traversed proof*.

By applying a cut with cut-formula ψ at every open leaf, every $\mathsf{F}\varphi$-traversed proof ρ corresponds to a GKe proof π, where on every branch of the proof there is at most one cut of rank $\mathsf{rank}(\mathsf{F}\varphi)$. Hence, transforming a $\mathsf{F}\varphi$-traversed proof to a traversed proof without open leaves corresponds to eliminating cuts of rank $\mathsf{rank}(\mathsf{F}\varphi)$.

Lemma 13. *Let π be a GKe proof with a critical cut of rank $n+1$ at the root, i.e. π is of the form*

$$\mathsf{cut}\ \frac{\overset{\pi_0}{\Sigma_0 \Rightarrow \Pi_0, \mathsf{F}\varphi} \qquad \overset{\pi_1}{\mathsf{F}\varphi^u, \Sigma_1 \Rightarrow \Pi_1}}{\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1}$$

where π_0 and π_1 are of cut-rank n, and the cut formula $\mathsf{F}\varphi$ has rank $n+1$. Then we can construct a proof π' of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ of cut-rank n.

Proof. Let ρ_I be the $\mathsf{F}\varphi$-traversed proof of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ consisting of an open leaf labelled by $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ together with $[\pi_0]\psi[\pi_1]$, this we will denote by

$$\frac{[\pi_0]\psi[\pi_1]}{\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1}$$

Starting from ρ_I we will inductively transform the traversed proof, until we end up with a traversed proof of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$, where all leaves are closed. This will be done as follows. Let ρ be a traversed proof.

If all leaves are closed, we are done. Otherwise, consider the leftmost open leaf v labelled by

$$\frac{[\pi_l]\psi[\pi_r]}{\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r}$$

Let the roots of π_l and π_r be labelled by the rules R_l and R_r, respectively. We transform ρ by a case distinction on R_l and R_r. We only show the crucial cases, the full proof can be found in Appendix B.

- If R_l is D^{x}, then π_l has the form

$$\mathsf{D}^{\mathsf{x}}\ \frac{\begin{matrix}\pi_l'\\ \Gamma_l \Rightarrow \Delta_l,\psi\end{matrix}}{\Gamma_l \Rightarrow \Delta_l,\psi}$$

  We proceed by making the following case distinction.

  - If there is a node c in ρ, that is an ancestor of v and labelled by $\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r$, such that the path from c to v is successful, then insert a D^{z} rule at c and let v be the discharged leaf $[\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r]^{\mathsf{z}}$ with fresh discharge token z.
  - Else, unfold π_l, i.e. let $\tilde{\pi}_l$ be the proof obtained from π_l' by replacing every discharged leaf labelled by x with π_l. We replace v by

$$\frac{[\tilde{\pi}_l]\psi[\pi_r]}{\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r}$$

- If R_r is D^{y}, then unfold π_r.
- If R_l is F_R and R_r is F_L, and π_l and π_r are of the forms

$$\mathsf{F}_R\ \frac{\begin{matrix}\pi_l'\\ \Gamma_l \Rightarrow \Delta_l,\varphi,\Diamond\mathsf{F}\varphi\end{matrix}}{\Gamma_l \Rightarrow \Delta_l,\mathsf{F}\varphi} \qquad \mathsf{F}_L\ \frac{\begin{matrix}\pi_r' & \pi_r^u\\ \Diamond\mathsf{F}\varphi^a,\Gamma_r \Rightarrow \Delta_r & \varphi^u,\Gamma_r \Rightarrow \Delta_r\end{matrix}}{\mathsf{F}\varphi^a,\Gamma_r \Rightarrow \Delta_r}$$

  then v is replaced by

$$\mathsf{cut}\ \frac{\begin{matrix}[\pi_l']\Diamond\mathsf{F}\varphi[\pi_r'] & \pi_r^u\\ \Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r,\varphi & \varphi^u,\Gamma_r \Rightarrow \Delta_r\end{matrix}}{\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r}$$

  where the introduced cut has rank n.

- If R_l is u or f of the form

$$\mathsf{R}_l\ \frac{\begin{matrix}\pi_l'\\ \Gamma_l' \Rightarrow \Delta_l,\psi\end{matrix}}{\Gamma_l \Rightarrow \Delta_l,\psi} \qquad \text{then } v \text{ is replaced by} \qquad \mathsf{R}_l\ \frac{\begin{matrix}[\pi_l']\psi[\pi_r]\\ \Gamma_l',\Gamma_r \Rightarrow \Delta_l,\Delta_r\end{matrix}}{\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r}$$

- If R_r is u or f of the form

$$\mathsf{R}_r\ \frac{\begin{matrix}\pi_r'\\ \psi^b,\Gamma_l' \Rightarrow \Delta_l\end{matrix}}{\psi^a,\Gamma_l \Rightarrow \Delta_l} \qquad \text{then } v \text{ is replaced by} \qquad \frac{[\pi_l]\psi[\pi_r']}{\Gamma_l,\Gamma_r \Rightarrow \Delta_l,\Delta_r}$$

- If R_l or R_r is any non-principal rule, we push the open leaf 'up', similar to how a cut is pushed away from the root.

Let ρ_i and ρ_j be traversed proofs. We write $\rho_i < \rho_j$, if ρ_j can be obtained from ρ_i by the above construction and $\rho_i \neq \rho_j$. If $\rho_i < \rho_j$, then ρ_i is a subproof of ρ_j, in the sense that ρ_j can be obtained from ρ_i by replacing some open leaves in ρ_i by traversed proofs and inserting nodes labelled by D. Thus, ρ_j consists of at least the nodes in ρ_i and we can identify nodes in ρ_i with nodes in ρ_j.
Let $\rho > \rho_I$ and let v be an open leaf in ρ labelled by

$$\frac{[\pi_l]\,\psi\,[\pi_r]}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

We can define a node $u_0(v) \in \pi_0$ with $\pi_l = \pi_{u_0(v)}$, $\mathsf{S}(u_0(v)) = \Gamma_l \Rightarrow \Delta_l$, ψ and an analogous node $u_1(v) \in \pi_1$. These definitions extend to nodes w below an open leaf v. Hence, nodes below an open leaf can be labelled by only finitely many distinct sequents.

Let τ be the path from the root of ρ to the open leaf v. Using the above definitions of $u_0(w)$ and $u_1(w)$ we obtain corresponding paths τ_0 in π_0 and τ_1 in π_1.

Combining these observations, we can find a bound on the height of open leaves: As there are only finitely many distinct sequents on τ, at some point an open leaf v is reached, such that an ancestor c in ρ is labelled by the same sequent. The path $\tau(v)$ from c to v in ρ corresponds to a path $\tau_0(v)$ in π_0. If $\tau(v)$ is of some certain length the path $\tau_0(v)$ is successful. This implies that the path $\tau(v)$ is successful as well and the leaf v will be closed. A more detailed proof with explicit bounds can be found in Appendix B.

In conclusion, as every constructed tree is finitely branching, after finitely many steps, a traversed proof ρ_T without open leaves is constructed. As every traversed proof without open leaves is a GKe proof, we are done. □

Example 14. Consider the following proof π containing a critical cut

$$\mathsf{cut}\ \frac{\overset{\pi_0}{\mathsf{F}\Diamond\Diamond p^u \Rightarrow \mathsf{F}\Diamond p} \qquad \overset{\pi_1}{\mathsf{F}\Diamond p^u \Rightarrow \mathsf{F}p}}{\mathsf{F}\Diamond\Diamond p^u \Rightarrow \mathsf{F}p}$$

where the two proofs π_0 and π_1 are

$$\Diamond_{\mathsf{F}_L}\ \frac{[\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}\Diamond p]^{\mathsf{x}}}{\Diamond\mathsf{F}\Diamond\Diamond p^f \Rightarrow \Diamond\mathsf{F}\Diamond p} \qquad \overline{\overline{\Diamond\Diamond p^u \Rightarrow \Diamond\mathsf{F}\Diamond p}}$$

$$a:\ \mathsf{F}_R\ \frac{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \Diamond p, \Diamond\mathsf{F}\Diamond p}{\mathsf{D}^{\mathsf{x}}\ \dfrac{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}\Diamond p}{\mathsf{u}\ \dfrac{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}\Diamond p}{\mathsf{F}\Diamond\Diamond p^u \Rightarrow \mathsf{F}\Diamond p}}}$$

$$\mathsf{F}_L\ \frac{[\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p]^{\mathsf{y}}}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\ \Diamond \qquad \overline{\overline{\Diamond p^u \Rightarrow \Diamond\mathsf{F}p}}$$

$$b:\ \mathsf{F}_R\ \frac{\mathsf{F}\Diamond p^f \Rightarrow p, \Diamond\mathsf{F}p}{\mathsf{D}^{\mathsf{y}}\ \dfrac{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}{\mathsf{u}\ \dfrac{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}{\mathsf{F}\Diamond p^u \Rightarrow \mathsf{F}p}}}$$

For the sake of readability we apply weakening implicitly and omit w rules in this example. The leaves marked by double lines can be proved easily without cuts or repeats.

Following the construction and notation of Lemma 13 we define ρ_I as above. In the first steps, the transformations for u in π_0 and π_1 are applied, then the resulting subproofs π_0' and π_1' get unfolded.

Adding the F_R rule in π_1 to ρ_I, results in the following traversed proof:

$$\dfrac{\dfrac{[\pi_a]\mathsf{F}\Diamond p[\pi_b]}{\mathsf{F}\Diamond\Diamond p^f \Rightarrow p, \Diamond\mathsf{F}p}\,\mathsf{F}_R}{\dfrac{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}p}{\mathsf{F}\Diamond\Diamond p^u \Rightarrow \mathsf{F}p}\,\mathsf{u}}$$

Now the rules R_l and R_r are F_R and F_L, respectively. Hence a cut with cut-formula $\Diamond p$ is introduced. Then the traversed proof gets further transformed into

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\pi_0']\mathsf{F}\Diamond p[\pi_1']}{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}p}\,\Diamond}{\Diamond\mathsf{F}\Diamond\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\,\mathsf{F}_L \quad \dfrac{}{\Diamond\Diamond p^u, \Rightarrow \Diamond\mathsf{F}p}}{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \Diamond p, \Diamond\mathsf{F}p}\,\mathsf{cut} \qquad \dfrac{}{\Diamond p^u \Rightarrow \Diamond\mathsf{F}p}}{\dfrac{\mathsf{F}\Diamond\Diamond p^f \Rightarrow p, \Diamond\mathsf{F}p}{\mathsf{F}\Diamond\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{F}_R}\,c:}{\mathsf{F}\Diamond\Diamond p^u \Rightarrow \mathsf{F}p}\,\mathsf{u}\;v:$$

The root of π_0' is labelled by D^{x} and the repeat condition for v is satisfied: The node c is labelled by the same sequent as v and the path from c to v is successful. Thus a D^{z} rule is inserted (with a fresh discharge token z) at c and v is discharged by z, which results in a GKe proof.

3.3 Unimportant cuts

Recall the set of component nodes of a node v in a derivation, written $\mathsf{comp}(v)$, defined in subsection 3.1.

Definition 15. Let π be a GKe derivation and v be a node in π. The *infinite unfolding of* $\mathsf{comp}(v)$ in π, written π^{*_v}, is obtained from π by recursively replacing every discharged leaf l, that is a component descendant of v, by π_l and removing nodes labelled by D^{x} whenever no discharged leaf is labelled by x.

Lemma 16. *Let π be a GKe proof of cut-rank n, such that all cuts of rank n are unimportant and in the root-cluster. Then we can transform π into a GKe proof π' of the same sequent with cut-rank $\leq n$, where all cuts of rank n are important.*

Proof. Let $\Gamma \Rightarrow \Delta$ be the sequent at the root r of π. For formulas φ and ψ, we write $\varphi \to_C \psi$ if ψ is either a direct subformula of φ or $\varphi \equiv \mathsf{F}\chi$ and $\psi \equiv \Diamond\mathsf{F}\chi$. The *closure* $\mathsf{Clos}(\Gamma \Rightarrow \Delta)$ of a sequent $\Gamma \Rightarrow \Delta$ is the least superset of $\Gamma \cup \Delta$ that is closed under $\to_C$. Let $m = |\mathsf{Clos}(\Gamma \Rightarrow \Delta)|$.

Let π^{*_r} be the infinite unfolding of $\mathsf{comp}(r)$ of π and let A be $\mathsf{comp}(r)$ in π^{*_r}. Note that A is infinite. We want to push the cuts (also the ones with cut-rank $< n$) occurring in A upwards in π^{*_r}, until the set of nodes A_0 of height $8^m + 1$ in A does not contain cuts.

Choose a root-most node v labelled by cut, take a lowest descendant v' of v, that is labelled by cut, such that both premises are not labelled by cut (that could also be v itself). This can always be found as π is a proof, hence a repeat cannot consist of cut-rules only. Apply a suitable cut reduction defined in Appendix A to v'. At some point this will be a non-principal cut reduction, because π is a proof and the cut-formula is out of focus: There are infinitely many F_L rules, where the principal formula is in focus on every infinite path in A. Thus the height of the cut-free sub-proof increases. Continue this process until A_0 is cut-free. All cuts of rank n, that where pushed outside of A, are important: If the cut is pushed

out of the cluster all formulas in the conclusion become out of focus (as π is minimally focused) and no cut reduction puts formulas in focus again. Because all cut-reductions defined in Appendix A preserve the cut-rank, all cuts have cut-rank $\leq n$.

As A_0 is cut-free, each formula in A_0 is in $\mathsf{Clos}(\Gamma \Rightarrow \Delta)$, where it could occur or not occur in Γ and Δ, and be in focus or out of focus in Γ. Thus, there are at most 8^m many distinct sequents in A_0 and on each branch in A_0 there is a node v such that an ancestor $c(v)$ of v is labelled by the same sequent. For each such branch choose the root-most such node v, insert a D^{x} rule at $c(v)$ and let v be a discharged leaf labelled by x with fresh discharge token x. All sequents in A_0 have a formula in focus. Because of Lemma 8 none of the cut-reductions affect formulas in focus, and F_L-rules where the principal formula is in focus, remain. Hence, the path from $c(v)$ to v is successful and we obtain a proof with cut-rank n, where all cuts of rank n are important. □

Example 17. Consider the following proof π containing one unimportant cut.

$$\dfrac{\dfrac{\dfrac{\dfrac{[\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p]^{\mathsf{x}}}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\,\Diamond \quad \dfrac{\dfrac{}{\Diamond p^u \Rightarrow \Diamond\mathsf{F}p}}{}}{\mathsf{F}\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\,\mathsf{F}_L \quad \dfrac{}{\Diamond\mathsf{F}p^u \Rightarrow \mathsf{F}p}}{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{cut}}{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{D}^{\mathsf{x}}$$

For the sake of readability, we apply weakening implicitly and omit w rules in this example. The leaves marked by double lines are in the next cluster and their (straightforward) proofs will be omitted. Following the construction described in Lemma 16 we push the cut upwards using the cut reductions from Appendix A. Here we only unfold the discharged leaf once, as it suffices in this example, in contrast to working with π^{*r} as in the proof of Lemma 16. First, a non-principal cut reduction for F_L is applied:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi}{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\,\Diamond \quad \dfrac{\dfrac{}{\Diamond\mathsf{F}p^u \Rightarrow p, \Diamond\mathsf{F}p}}{\Diamond\mathsf{F}p^u \Rightarrow \mathsf{F}p}\,\mathsf{F}_R}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{cut} \quad \dfrac{\dfrac{}{\Diamond p^u \Rightarrow \Diamond\mathsf{F}p} \quad \dfrac{}{\Diamond\mathsf{F}p^u \Rightarrow \mathsf{F}p}}{\Diamond p^u \Rightarrow \mathsf{F}p}\,\mathsf{cut}}{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{F}_L$$

The cut in the right subproof is pushed out of the cluster. In the left subproof, principal cut-reductions for F_R and $\Diamond$ are applied:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi}{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \Diamond\mathsf{F}p}\,\Diamond \quad \dfrac{}{\Diamond\mathsf{F}p^u \Rightarrow \mathsf{F}p}}{v:\quad \dfrac{\dfrac{\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow p, \Diamond\mathsf{F}p}\,\Diamond}{\Diamond\mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{F}_R}\,\mathsf{cut} \quad \dfrac{\dfrac{}{\Diamond p^u \Rightarrow \Diamond\mathsf{F}p} \quad \dfrac{}{\Diamond\mathsf{F}p^u \Rightarrow \mathsf{F}p}}{\Diamond p^u \Rightarrow \mathsf{F}p}\,\mathsf{cut}}{c:\quad \mathsf{F}\Diamond p^f \Rightarrow \mathsf{F}p}\,\mathsf{F}_L$$

The node v below the cut is labelled by the same sequent as its ancestor c and the path from c to v is successful. Thus a D^{y} is inserted at c and v is discharged by y. This results in a GKe proof, where the only remaining cut is important.

3.4 Main theorem

We want to prove cut-elimination for GKe by induction on the cut-rank, hence it suffices to transform any proof π of cut-rank $n+1$ to a proof π' of cut-rank n.

Lemma 18 (Reduction Lemma)**.** *Let π be a proof of cut-rank $n+1$. Then we can transform π into a* GKe *proof π' of cut-rank n of the same sequent.*

Proof. By induction on the number of clusters in π with unimportant cuts of rank $n+1$ with a sub-induction on the number of clusters in π with important cuts of rank $n+1$.

Let π_0 be a subproof of π, where all cuts with rank $n+1$ are in the root-cluster. If the root-cluster is trivial, there is one important cut at the root of π_0. Otherwise, all cuts in the root-cluster of π_0 are unimportant. In the first case, Lemma 11 transforms π_0 to π_0', where all cuts of rank $n+1$ are critical. Those can then be reduced using Lemma 13, which yields a proof π_1 with cut-rank n. In the second case, Lemma 16 yields a proof π_1 with cut-rank $n+1$, where all cuts of rank $n+1$ are important. In both cases, we substitute π_0 by π_1 in π and obtain a proof π', where we can apply the induction hypothesis. □

Theorem 19 (Cut elimination)**.** *We can transform every* GKe *proof π into a cut-free* GKe *proof π' of the same sequent.*

Proof. By induction on the cut-rank of π using Lemma 18. □

4 Conclusion

We have presented a syntactic cut-elimination procedure for the cyclic proof system GKe. To our knowledge, this is the first cut-elimination method that works directly on cyclic proofs without a detour through infinitary proofs. We anticipate that the introduced approach generalises to other annotated, cyclic proof systems for fragments of the modal μ-calculus. Two natural candidates are Propositional Dynamic Logic and the alternation-free μ-calculus. The former has a similar modal structure as MLe; the latter is known to admit a focused cyclic proof system [8]. In both cases important and unimportant cuts can be defined as in this paper, allowing an analogous cut elimination process for unimportant cuts. Complications arise in the treatment of important cuts, especially if formulas in focus occur in multiple premises of a rule or multiple formulas in a sequent are in focus. A potential mitigation would be to employ a multi-cut rule.

References

[1] Julian Bradfield & Colin Stirling (2007): *Modal mu-Calculi*. In Patrick Blackburn, Johan Van Benthem & Frank Wolter, editors: *Handbook of Modal Logic*, *Studies in Logic and Practical Reasoning* 3, Elsevier, pp. 721–756, doi:10.1016/S1570-2464(07)80015-2.

[2] James Brotherston (2006): *Sequent Calculus Proof Systems for Inductive Definitions*. Ph.D. thesis, University of Edinburgh. Available at https://era.ed.ac.uk/handle/1842/1458.

[3] Anupam Das & Damien Pous (2018): *Non-Wellfounded Proof Theory For (Kleene+Action)(Algebras+Lattices)*. In Dan Ghica & Achim Jung, editors: *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 119, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 19:1–19:18, doi:10.4230/LIPIcs.CSL.2018.19. Available at http://drops.dagstuhl.de/opus/volltexte/2018/9686.

[4] Maurice Dekker, Johannes Kloibhofer, Johannes Marti & Yde Venema (2023): *Proof Systems for the Modal μ-Calculus Obtained by Determinizing Automata*. In: *TABLEAUX 2023, Prague, Czech Republic*, *Lecture Notes in Computer Science* 14278, Springer, pp. 242–259, doi:10.1007/978-3-031-43513-3_14.

[5] Jérôme Fortier & Luigi Santocanale (2013): *Cuts for Circular Proofs: Semantics and Cut-elimination*. In: *Computer Science Logic 2013 (CSL 2013)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 23, pp. 248–262, doi:10.4230/LIPIcs.CSL.2013.248.

[6] Natthapong Jungteerapanich (2010): *Tableau Systems for the Modal μ-Calculus*. Ph.D. thesis, University of Edinburgh. Available at http://hdl.handle.net/1842/4208.

[7] Graham E. Leigh & Dominik Wehr (2024): *From GTC to Image 1: Generating reset proof systems from cyclic proof systems*. *Annals of Pure and Applied Logic* 175(10), p. 103485, doi:10.1016/j.apal.2024.103485.

[8] Johannes Marti & Yde Venema (2021): *A Focus System for the Alternation-Free μ-Calculus*. In: *TABLEAUX 2021, Proceedings*, Lecture Notes in Computer Science, Springer, pp. 371–388, doi:10.1007/978-3-030-86059-2_22.

[9] Damian Niwinski & Igor Walukiewicz (1996): *Games for the mu-Calculus*. *Theor. Comput. Sci.* 163(1&2), pp. 99–116, doi:10.1016/0304-3975(95)00136-0.

[10] Jan Rooduijn (2021): *Cyclic Hypersequent Calculi for Some Modal Logics with the Master Modality*. In: *TABLEAUX 2021*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 354–370, doi:10.1007/978-3-030-86059-2_21.

[11] Luigi Santocanale (2002): *A Calculus of Circular Proofs and Its Categorical Semantics*. In: *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 357–371, doi:10.1007/3-540-45931-6_25.

[12] Alexis Saurin (2023): *A Linear Perspective on Cut-Elimination for Non-wellfounded Sequent Calculi with Least and Greatest Fixed-Points*. In: *TABLEAUX 2023, Prague, Czech Republic*, *Lecture Notes in Computer Science* 14278, Springer, pp. 203–222, doi:10.1007/978-3-031-43513-3_12.

[13] Yury Savateev & Daniyar Shamkanov (2020): *Non-well-founded Proofs for the Grzegorcyk Modal Logic*. *The Review of Symbolic Logic* 14(1), pp. 22–50, doi:10.1017/s1755020319000510.

[14] Kurt Schütte (1977): *Proof Theory*. Springer-Verlag, Berlin, Heidelberg, doi:10.1007/978-3-642-66473-1.

[15] Colin Stirling (2014): *A Tableau Proof System with Names for Modal Mu-calculus*. In: *HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, *EPiC Series in Computing* 42, EasyChair, pp. 306–318, doi:10.29007/lwqm.

A Appendix: Cut Reductions

For readability we state the cut reductions for a simplified cut rule, where $\Gamma_l = \Gamma_r$ and $\Delta_l = \Delta_r$; this can be generalised in the obvious way.

A.1 Principal cut reductions

$$\mathsf{F}_R\ \dfrac{\mathsf{cut}\ \dfrac{\dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi, \Diamond \mathsf{F}\varphi}}{\Gamma \Rightarrow \Delta, \mathsf{F}\varphi} \quad \mathsf{F}_L\ \dfrac{\dfrac{\overset{\pi_1}{\Diamond \mathsf{F}\varphi^u, \Gamma \Rightarrow \Delta} \quad \overset{\pi_2}{\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{F}\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

$$\longrightarrow \quad \mathsf{cut}\ \dfrac{\dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi, \Diamond \mathsf{F}\varphi} \quad \overset{\pi_1}{\Diamond \mathsf{F}\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{cut}\ \dfrac{\Gamma \Rightarrow \Delta, \varphi \quad \overset{\pi_2}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}}$$

$$\Diamond\ \dfrac{\mathsf{cut}\ \dfrac{\dfrac{\overset{\pi_0}{\gamma^a \Rightarrow \Delta, \varphi}}{\Diamond \gamma^a \Rightarrow \Diamond \Delta, \Diamond \varphi} \quad \Diamond\ \dfrac{\overset{\pi_1}{\varphi^u \Rightarrow \Delta}}{\Diamond \varphi^u \Rightarrow \Diamond \Delta}}{\Diamond \gamma^a \Rightarrow \Diamond \Delta}$$

$$\longrightarrow \quad \mathsf{cut}\ \dfrac{\dfrac{\overset{\pi_0}{\gamma^a \Rightarrow \Delta, \varphi} \quad \overset{\pi_1}{\varphi^u \Rightarrow \Delta}}{\Diamond\ \dfrac{\gamma^a \Rightarrow \Delta}{\Diamond \gamma^a \Rightarrow \Diamond \Delta}}$$

$$\wedge_R\ \dfrac{\mathsf{cut}\ \dfrac{\dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \quad \overset{\pi_1}{\Gamma \Rightarrow \Delta, \psi}}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi} \quad \wedge_L\ \dfrac{\overset{\pi_2}{\varphi^u, \psi^u, \Gamma \Rightarrow \Delta}}{\varphi \wedge \psi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

$$\longrightarrow \quad \mathsf{cut}\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \quad \mathsf{cut}\ \dfrac{\overset{\pi_1}{\Gamma \Rightarrow \Delta, \psi} \quad \overset{\pi_2}{\varphi^u, \psi^u, \Gamma \Rightarrow \Delta}}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

$$\neg_R\ \dfrac{\dfrac{\overset{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta, \neg \varphi} \quad \neg_L\ \dfrac{\overset{\pi_1}{\Gamma \Rightarrow \Delta, \varphi}}{\neg \varphi^u, \Gamma \Rightarrow \Delta}\ \mathsf{cut}}{\Gamma \Rightarrow \Delta}$$

$$\longrightarrow \quad \mathsf{cut}\ \dfrac{\overset{\pi_1}{\Gamma \Rightarrow \Delta, \varphi} \quad \overset{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

A.2 Trivial principal cut reductions

$$\mathsf{cut}\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, p} \quad p^u \Rightarrow p}{\Gamma \Rightarrow \Delta, p} \quad \longrightarrow \quad \overset{\pi_0}{\Gamma \Rightarrow \Delta, p}$$

$$\mathsf{cut}\ \dfrac{p^u \Rightarrow p \quad \overset{\pi_1}{p^u, \Gamma \Rightarrow \Delta}}{p^u, \Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \overset{\pi_1}{p^u, \Gamma \Rightarrow \Delta}$$

$$\mathsf{cut}\ \dfrac{\mathsf{w}_R\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \overset{\pi_0}{\Gamma \Rightarrow \Delta}$$

$$\mathsf{cut}\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \qquad \mathsf{w}_L\ \dfrac{\overset{\pi_1}{\Gamma \Rightarrow \Delta}}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \overset{\pi_1}{\Gamma \Rightarrow \Delta}$$

A.3 Cut reductions for D, u and f

We push u and f rules 'upwards' away from the root and unfold D rules. The presented reductions are analogous, if the right premise of the cut is labelled by D, u or f.

$$v:\ \mathsf{cut}\ \dfrac{\mathsf{D}^{\mathsf{x}}\ \dfrac{\overset{\pi_0}{\Gamma \Rightarrow \Delta, \varphi}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \mathsf{cut}\ \dfrac{\overset{\pi_0'}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

where π_0' is obtained from π_0 by replacing every discharged leaf labelled by x with π_v, where v is the left premise of the cut rule.[2]

$$v:\ \mathsf{cut}\ \dfrac{\mathsf{u}\ \dfrac{\mathsf{D}^{\mathsf{x}}\ \dfrac{\overset{\pi_0}{\Gamma' \Rightarrow \Delta, \varphi}}{\Gamma' \Rightarrow \Delta, \varphi}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \mathsf{cut}\ \dfrac{\overset{\pi_0'}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

where π_0' is obtained from π_0 by (i) unfocusing sequents up to D rules and leaves labelled by x and (ii) replacing every discharged leaf labelled by x with subproof π_v, where v is the left premise of the cut rule.

$$\dfrac{\mathsf{f}\ \dfrac{\overset{\pi_0}{\Gamma^u \Rightarrow \Delta, \varphi}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \mathsf{f}\ \dfrac{\overset{\pi_1}{\varphi^u, \Gamma^u \Rightarrow \Delta}}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}\ \mathsf{cut} \quad \longrightarrow \quad \mathsf{cut}\ \dfrac{\overset{\pi_0}{\Gamma^u \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma^u \Rightarrow \Delta}}{\mathsf{f}\ \dfrac{\Gamma^u \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}}$$

If the rule at the root of π_1 is different to f we do the following:

$$\mathsf{cut}\ \dfrac{\mathsf{f}\ \dfrac{\mathsf{u}\ \dfrac{\mathsf{D}^{\mathsf{x}}\ \dfrac{\overset{\pi_0}{\Gamma' \Rightarrow \Delta, \varphi}}{\Gamma' \Rightarrow \Delta, \varphi}}{\Gamma^u \Rightarrow \Delta, \varphi}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta} \quad \longrightarrow \quad \mathsf{cut}\ \dfrac{\mathsf{f}\ \dfrac{\overset{\pi_0'}{\Gamma^u \Rightarrow \Delta, \varphi}}{\Gamma \Rightarrow \Delta, \varphi} \qquad \overset{\pi_1}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma \Rightarrow \Delta}$$

[2] Here and in the following cut reductions discharge tokens y are replaced by fresh discharge tokens, whenever a D^{y} rule is duplicated.

where π_0' is defined as above. For a rule R different from u we proceed as follows.

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma_1 \Rightarrow \Delta_1, \varphi} \quad \cdots \quad \dfrac{\pi_n}{\Gamma_n \Rightarrow \Delta_n, \varphi}}{\mathsf{R}\ \dfrac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma \Rightarrow \Delta, \varphi}\,\mathsf{f}} \qquad \dfrac{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{cut}\ \Gamma \Rightarrow \Delta} \longrightarrow \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma_1 \Rightarrow \Delta_1, \varphi}\,\mathsf{f} \quad \cdots \quad \dfrac{\pi_n}{\Gamma_n \Rightarrow \Delta_n, \varphi}\,\mathsf{f}}{\mathsf{R}\ \Gamma_1' \Rightarrow \Delta_1, \varphi \quad \cdots \quad \Gamma_n' \Rightarrow \Delta_n, \varphi} \qquad \dfrac{\Gamma \Rightarrow \Delta, \varphi \qquad \dfrac{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{cut}}}{\Gamma \Rightarrow \Delta}$$

A.4 Non-principal cut-reductions for a rule R different from $\Diamond$, u, f and D

$$\dfrac{\dfrac{\dfrac{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \qquad \dfrac{\dfrac{\pi_1}{\varphi^u, \Gamma_1 \Rightarrow \Delta_1} \quad \cdots \quad \dfrac{\pi_n}{\varphi^u, \Gamma_n \Rightarrow \Delta_n}}{\varphi^u, \Gamma \Rightarrow \Delta}\,\mathsf{R}}{\Gamma \Rightarrow \Delta}\,\mathsf{cut} \longrightarrow \dfrac{\dfrac{\dfrac{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \qquad \dfrac{\pi_1}{\varphi^u, \Gamma_1 \Rightarrow \Delta_1}}{\Gamma_1, \Gamma \Rightarrow \Delta_1, \Delta}\,\mathsf{cut} \quad \cdots \quad \dfrac{\dfrac{\pi_0}{\Gamma \Rightarrow \Delta, \varphi} \qquad \dfrac{\pi_n}{\varphi^u, \Gamma_n \Rightarrow \Delta_n}}{\Gamma_n, \Gamma \Rightarrow \Delta_n, \Delta}\,\mathsf{cut}}{\Gamma \Rightarrow \Delta}\,\mathsf{R}$$

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma_1 \Rightarrow \Delta_1, \varphi} \quad \cdots \quad \dfrac{\pi_n}{\Gamma_n \Rightarrow \Delta_n, \varphi}}{\mathsf{R}\ \Gamma \Rightarrow \Delta, \varphi} \qquad \dfrac{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\mathsf{cut}\ \Gamma \Rightarrow \Delta} \longrightarrow \dfrac{\dfrac{\dfrac{\pi_1}{\Gamma_1 \Rightarrow \Delta_1, \varphi} \qquad \dfrac{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma_1, \Gamma \Rightarrow \Delta_1, \Delta}\,\mathsf{cut} \quad \cdots \quad \dfrac{\dfrac{\pi_n}{\Gamma_n \Rightarrow \Delta_n, \varphi} \qquad \dfrac{\pi_0}{\varphi^u, \Gamma \Rightarrow \Delta}}{\Gamma_n, \Gamma \Rightarrow \Delta_n, \Delta}\,\mathsf{cut}}{\Gamma \Rightarrow \Delta}\,\mathsf{R}$$

Note, the rule R in this case may also be an instance of cut.

B Appendix: Important Cuts

The *size* of a GKe proof π is the number of nodes in π. For a path τ in π we call a path τ' a *subpath* of τ, if τ can be written as $\tau = \sigma_0\tau\sigma_1$ for some paths σ_0, σ_1 in π.

Lemma B.1. *Let τ be a path through a* GKe *proof π of size n. If $l(\tau) \geq k \cdot n$, then there is a cluster S of π of size n_S such that there is a subpath τ' of τ in S with $l(\tau') \geq k \cdot n_S$.*

Proof. Let $S_1, \ldots, S_m$ be the clusters of π. Then $|\pi| = \sum_{j=1}^{m} |S_j|$. The lemma follows, as for $i \neq j$ there is either no path from S_i to S_j or no path from S_j to S_i. □

Lemma B.2. *Let S be a non-trivial cluster of size n in a* GKe *proof π and $\tau = v_1 v_2 \ldots$ be a path τ in S. Let $l(\tau)$ be the* length *of the path τ. Then,*

1. *if $l(\tau) \geq n$, then τ is successful,*
2. *if $l(\tau) \geq n$, then there is a companion node on τ,*
3. *if $l(\tau) \geq n$, then there is a node labelled by $\Diamond$ on τ,*
4. *if $l(\tau) \geq 2 \cdot k \cdot n$ then there are k indices $1 \leq m_1 < \cdots < m_k \leq 2 \cdot k \cdot n$ such that $v_{m_1}, \ldots, v_{m_k}$ are companion nodes and the paths $v_{m_i} v_{m_i+1} \ldots v_{m_j}$ are successful for all $i < j$.*

Proof. Every node of τ is in S, hence it has a formula in focus. Thus for 1 we only have to show that τ passes through an application of F_L, where the principal formula is in focus. If every node of S is visited, then 1, 2 and 3 are satisfied. Otherwise, there has to be a node that is visited twice. This is only possible if there is a discharged leaf u such that every node of $\tau(u)$ is in τ. Thus 1, 2 and 3 are satisfied. 4 follows by combining 2 and 1. □

We now give a detailed proof of Lemma 13.

Lemma 13. *Let π be a* GKe *proof with a critical cut of rank $n+1$ at the root, i.e. π is of the form*

$$\mathsf{cut}\ \dfrac{\overset{\pi_0}{\Sigma_0 \Rightarrow \Pi_0, \mathsf{F}\varphi} \qquad \overset{\pi_1}{\mathsf{F}\varphi^u, \Sigma_1 \Rightarrow \Pi_1}}{\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1}$$

where π_0 and π_1 are of cut-rank n, and the cut formula $\mathsf{F}\varphi$ has rank $n+1$. Then we can construct a proof π' of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ of cut-rank n.

Proof. Let ρ_I be the $\mathsf{F}\varphi$-traversed proof of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ consisting of an open leaf labelled by $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$ together with $[\pi_0]\psi[\pi_1]$, this we will denote by

$$\dfrac{[\pi_0]\psi[\pi_1]}{\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1}$$

Starting from ρ_I we will inductively transform the traversed proof, until we end up with a traversed proof of $\Sigma_0, \Sigma_1 \Rightarrow \Pi_0, \Pi_1$, where all leaves are closed. This will be done as follows: Let ρ be a traversed proof. If all leaves are closed we are done. Otherwise consider the leftmost open leaf v labelled by

$$\dfrac{[\pi_l]\psi[\pi_r]}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

Let the roots of π_l and π_r be labelled by the rules R_l and R_r, respectively. We transform ρ by a case distinction on R_l and R_r.

- If R_l is $\mathsf{D}^\times$, then π_l has the form

$$\dfrac{\overset{\pi'_l}{\Gamma_l \Rightarrow \Delta_l, \psi}}{\Gamma_l \Rightarrow \Delta_l, \psi}\ \mathsf{D}^\times$$

We proceed by making the following case distinction.

- If there is a node c in ρ, that is an ancestor of v and labelled by $\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r$, such that the path from c to v is successful, then insert a D^{z} rule at c and let v be the discharged leaf $[\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r]^{\mathsf{z}}$ with fresh discharge token z.
- Else, unfold π_l, i.e. let $\tilde{\pi}_l$ be the proof obtained from π_l' by replacing every discharged leaf labelled by x with π_l. We replace v by

$$\dfrac{[\tilde{\pi}_l]\ \psi\ [\pi_r]}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

- If R_l is F_R with principal formula $\mathsf{F}\varphi$ we make a case distinction on R_r:
  - If R_r is F_L with principal formula $\mathsf{F}\varphi$, the proofs π_l and π_r have the form

$$\mathsf{F}_R\ \dfrac{\begin{matrix}\pi_l' \\ \Gamma_l \Rightarrow \Delta_l, \varphi, \Diamond\mathsf{F}\varphi\end{matrix}}{\Gamma_l \Rightarrow \Delta_l, \mathsf{F}\varphi} \qquad \mathsf{F}_L\ \dfrac{\begin{matrix}\pi_r' \\ \Diamond\mathsf{F}\varphi^a, \Gamma_r \Rightarrow \Delta_r\end{matrix} \qquad \begin{matrix}\pi_r^u \\ \varphi^u, \Gamma_r \Rightarrow \Delta_r\end{matrix}}{\mathsf{F}\varphi^a, \Gamma_r \Rightarrow \Delta_r}$$

    and v is replaced by

$$\mathsf{cut}\ \dfrac{\begin{matrix}[\pi_l']\Diamond\mathsf{F}\varphi[\pi_r'] \\ \Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r, \varphi\end{matrix} \qquad \begin{matrix}\pi_r^u \\ \varphi^u, \Gamma_r \Rightarrow \Delta_r\end{matrix}}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

    where the introduced cut has rank $\mathsf{rank}(\varphi)$.
  - If R_r is u of the form

$$\mathsf{u}\ \dfrac{\begin{matrix}\pi_r' \\ \psi^a, \Gamma_r' \Rightarrow \Delta_r\end{matrix}}{\psi^u, \Gamma_r \Rightarrow \Delta_r} \qquad \text{then } v \text{ is replaced by} \qquad \dfrac{[\pi_l]\ \psi\ [\pi_r']}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

  - If R_r is f of the form

$$\mathsf{f}\ \dfrac{\begin{matrix}\pi_r' \\ \psi^u, \Gamma_r^u \Rightarrow \Delta_r\end{matrix}}{\psi^a, \Gamma_r \Rightarrow \Delta_r} \qquad \text{then } v \text{ is replaced by} \qquad \dfrac{[\pi_l]\ \psi\ [\pi_r']}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

  - If R_r is D^{y} of the form

$$\mathsf{D}^{\mathsf{y}}\ \dfrac{\begin{matrix}\pi_r' \\ \psi^a, \Gamma_r \Rightarrow \Delta_r\end{matrix}}{\psi^a, \Gamma_r \Rightarrow \Delta_r}$$

    then unfold π_r, i.e. let $\tilde{\pi}_r$ be the proof obtained from π_r' by replacing every discharged leaf labelled by y with π_r.[3] We replace v by

$$\dfrac{[\pi_l]\ \psi\ [\tilde{\pi}_r]}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

[3] Discharge tokens z are replaced by fresh discharge tokens, whenever a D^{z} rule is duplicated.

– If R_r is w of the form

$$\mathsf{w}\ \dfrac{\begin{array}{c}\pi'_r\\ \Gamma'_r \Rightarrow \Delta_r\end{array}}{\psi^u, \Gamma'_r \Rightarrow \Delta_r} \qquad \text{then } v \text{ is replaced by} \qquad \mathsf{w}\ \dfrac{\begin{array}{c}\pi'_r\\ \Gamma'_r \Rightarrow \Delta_r\end{array}}{\Gamma_r \Rightarrow \Delta_r}\ \mathsf{u}\ \dfrac{}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

– If R_r is any other rule, then the proof has the form

$$\mathsf{R}\ \dfrac{\begin{array}{c}\pi^1_r\\ \psi^a, \Gamma^1_r \Rightarrow \Delta^1_r\end{array} \quad \dots \quad \begin{array}{c}\pi^n_r\\ \psi^a, \Gamma^n_r \Rightarrow \Delta^n_r\end{array}}{\psi^a, \Gamma_r \Rightarrow \Delta_r}$$

and v is replaced by

$$\mathsf{R}\ \dfrac{\begin{array}{c}[\pi_l]\psi[\pi^1_r]\\ \Gamma_l, \Gamma^1_r \Rightarrow \Delta_l, \Delta^1_r\end{array} \quad \dots \quad \begin{array}{c}[\pi_l]\psi[\pi^n_r]\\ \Gamma_l, \Gamma^n_r \Rightarrow \Delta_l, \Delta^n_r\end{array}}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

- If R_l is $\Diamond$, then we make a case distinction on R_r:
  – If R_r is $\Diamond$, the proofs have the form

$$\Diamond\ \dfrac{\begin{array}{c}\pi'_l\\ \chi^a \Rightarrow \Delta_l, \mathsf{F}\varphi\end{array}}{\Diamond\chi^a \Rightarrow \Diamond\Delta_l, \Diamond\mathsf{F}\varphi} \qquad \Diamond\ \dfrac{\begin{array}{c}\pi'_r\\ \mathsf{F}\varphi^b \Rightarrow \Delta_r\end{array}}{\Diamond\mathsf{F}\varphi^b \Rightarrow \Diamond\Delta_r}$$

and v is replaced by

$$\Diamond\ \dfrac{\begin{array}{c}[\pi'_l]\mathsf{F}\varphi[\pi'_r]\\ \chi^a \Rightarrow \Delta_l, \Delta_r\end{array}}{\Diamond\chi^a \Rightarrow \Diamond\Delta_l, \Diamond\Delta_r}$$

  – If R_r is any other rule, we do the same as in the previous case.

- If R_l is u of the form

$$\mathsf{u}\ \dfrac{\begin{array}{c}\pi'_l\\ \Gamma'_l \Rightarrow \Delta_l, \psi\end{array}}{\Gamma_l \Rightarrow \Delta_l, \psi} \qquad \text{then } v \text{ is replaced by} \qquad \mathsf{u}\ \dfrac{\begin{array}{c}[\pi'_l]\psi[\pi_r]\\ \Gamma'_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r\end{array}}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

- If R_l is f of the form

$$\mathsf{f}\ \dfrac{\begin{array}{c}\pi'_l\\ \Gamma^u_l \Rightarrow \Delta_l, \psi\end{array}}{\Gamma_l \Rightarrow \Delta_l, \psi} \qquad \text{then } v \text{ is replaced by} \qquad \mathsf{f}\ \dfrac{\begin{array}{c}[\pi'_l]\psi[\pi_r]\\ \Gamma^u_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r\end{array}}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

- If R_l is w of the form

$$\mathsf{w}\ \dfrac{\begin{array}{c}\pi'_l\\ \Gamma_l \Rightarrow \Delta_l\end{array}}{\Gamma_l \Rightarrow \Delta_l, \psi} \qquad \text{then } v \text{ is replaced by} \qquad \mathsf{w}\ \dfrac{\begin{array}{c}\pi'_l\\ \Gamma_l \Rightarrow \Delta_l\end{array}}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

- If R_l is any other rule, π_l has the form

$$\mathsf{R}\ \frac{\pi_l^1 \qquad\qquad \pi_l^n}{\Gamma_l^1 \Rightarrow \Delta_l^1, \psi \quad \dots \quad \Gamma_l^n \Rightarrow \Delta_l^n, \psi}{\Gamma_l \Rightarrow \Delta_l, \psi}$$

  and v is replaced by

$$\mathsf{R}\ \frac{[\pi_l^1]\psi[\pi_r] \qquad\qquad [\pi_l^n]\psi[\pi_r]}{\Gamma_l^1, \Gamma_r \Rightarrow \Delta_l^1, \Delta_r \quad \dots \quad \Gamma_l^n, \Gamma_r \Rightarrow \Delta_l^n, \Delta_r}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

Let ρ_i and ρ_j be traversed proofs. We write $\rho_i < \rho_j$ if ρ_j can be obtained from ρ_i by the above construction and $\rho_i \neq \rho_j$. It holds that $<$ is irreflexive, antisymmetric and transitive. Moreover, if $\rho_i < \rho_j$, then ρ_i is a subproof of ρ_j, in the sense that ρ_j can be obtained from ρ_i by replacing some open leaves in ρ_i by traversed proofs and inserting nodes labelled by D. Thus, ρ_j consists of at least the nodes in ρ_i and we can identify nodes in ρ_i with nodes in ρ_j.

From now on, whenever we speak about a traversed proof ρ we mean a traversed proof $\rho > \rho_I$.

Let $\rho > \rho_I$ and let v be an open leaf in ρ labelled by

$$\frac{[\pi_l]\psi[\pi_r]}{\Gamma_l, \Gamma_r \Rightarrow \Delta_l, \Delta_r}$$

We first define nodes $u_0(v) \in \pi_0$ and $u_1(v) \in \pi_1$ with

1. $\pi_l = \pi_{u_0(v)}$ and $\pi_r = \pi_{u_1(v)}$,
2. $\mathsf{S}(u_0(v)) = \Gamma_l \Rightarrow \Delta_l, \psi$ and $\mathsf{S}(u_1(v)) = \psi^a, \Gamma_r' \Rightarrow \Delta_r$,

The nodes $u_0(v)$ and $u_1(v)$ are defined by recursion on the construction. For ρ_I define $u_0(v)$ to be the root of π_0 and $u_1(v)$ to be the root of π_1. For the recursion step we follow the case distinction. For example, let ρ be a traversed proof with leftmost open leaf v, where the last applied rules in π_l and π_r are $\Diamond$, respectively. Let ρ' be obtained from ρ in one step with leftmost open leaf v'. Then $u_0(v')$ is the child of $u_0(v)$ and $u_1(v')$ is the child of $u_1(v)$.

This definition extends to nodes w in a traversed proof ρ below an open leaf. For such nodes w and w' it moreover holds, that if w is the parent of w', then either

1. $u_0(w) = u_0(w')$,
2. $u_0(w)$ is the parent of $u_0(w')$ or
3. $u_0(w)$ is an ancestor of $u_0(w')$, where all but one node on the ancestor path are labelled by D.

The same holds for the nodes $u_1(w)$ and $u_1(w')$.

Let τ be the path from the root of ρ to the open leaf v. By the above definition we can define corresponding paths τ_0 in π_0 and τ_1 in π_1.

Let n_0 be the size of π_0 and n_1 be the size of π_1. By the above argumentation nodes in τ can only be labelled by at most $n_0 \cdot n_1$ distinct sequents. Moreover, if $l(\tau) = k \cdot n_1$, then $l(\tau_0) \geq k$. This holds as in every step of the construction either π_l or π_r is transformed. Whenever π_l is transformed case 1 above cannot be the case. But if $l(\tau) \geq n_1$, π_l has to be transformed, as otherwise $l(\tau_1) \geq n_1$. Due to Lemma B.1 and B.2.3 there has to be a $\Diamond$ rule on τ_1, in which case π_l gets transformed as well.

We claim that every open leaf in a traversed proof ρ has height at most $4 \cdot n_0^2 \cdot n_1^2$. For suppose that v is an open leaf of height more than $4 \cdot n_0^2 \cdot n_1^2$. Let τ be the path from the root of ρ to v and let τ_0 be the corresponding path in π_0. It holds that $l(\tau_0) \geq 4 \cdot n_0^2 \cdot n_1$. Lemma B.1 gives a cluster S with size n_S of π_0 and a subpath τ_S in S of τ_0 such that $l(\tau_S) \geq 4 \cdot n_S \cdot n_0 \cdot n_1$. Using Lemma B.2.4 we obtain $2 \cdot n_0 \cdot n_1$ companion nodes $c_1, c_2, \ldots$ in S. Hence there are as many traversed proofs with leftmost open leaves $v_1, v_2, \ldots$ with $u_0(v_j) = c_j$ for all j. This means that there are $2 \cdot n_0 \cdot n_1$ traversed proofs $\rho_1 < \rho_2 < \cdots < \rho$, with left-most open leaves $v_1, v_2, \ldots$, which are *repeat leaves*, i.e. where π_l is of the form

$$\mathsf{D}^{\times}\ \dfrac{\begin{array}{c}\pi_l' \\ \Gamma_l \Rightarrow \Delta_l, \psi\end{array}}{\Gamma_l \Rightarrow \Delta_l, \psi}$$

By the above argumentation the nodes $v_1, v_2, \ldots$ are labelled by at most $n_0 \cdot n_1$ many distinct sequents. Hence there are $i < j$ such that $\mathsf{S}(v_i) = \mathsf{S}(v_j)$. Due to the choice of the companion nodes in Lemma B.2.4 the path from c_i to c_j in τ_S is successful. Note that, if $u_0(w)$ is labelled by F_L, where the principal formula is in focus, then so is w. Because all sequents in τ_S have a formula in focus, so does the subpath from v_i to v_j of τ. Hence, the path from v_i to v_j is successful and the leaf gets closed in the next step of transforming ρ_j. This contradicts the fact that v in ρ is an open leaf of height more than $4 \cdot n_0^2 \cdot n_1^2$.

In conclusion, as every constructed tree is finitely branching, after finitely many steps a traversed proof ρ_T without open leaves is constructed. As every traversed proof without open leaves is a GKe proof we have shown the lemma. □

Ruitenburg's Theorem Mechanized and Contextualized

Tadeusz Litak*
University of Naples Federico II
tadeusz.litak@unina.it

In 1984, Wim Ruitenburg published a surprising result about periodic sequences in intuitionistic propositional calculus (IPC). The property established by Ruitenburg naturally generalizes local finiteness; recall that intuitionistic logic is not locally finite, even in a single variable. One of the two main goals of this note is to illustrate that most "natural" non-classical logics failing local finiteness also do not enjoy the periodic sequence property. IPC is quite unique in separating these properties. The other goal of this note is to present a Coq formalization of Ruitenburg's heavily syntactic proof. Apart from ensuring its correctness, the formalization allows extraction of a program providing a certified implementation of Ruitenburg's algorithm.

1 Introduction

In 1984, Wim Ruitenburg [39] published a surprising result about periodic sequences in intuitionistic propositional calculus (IPC). For quite a while, the result seemed relatively neglected despite having been discussed, e.g., in Humberstone's monograph on logical connectives [23], and used in the work of the late Sergey Mardaev [26, 27, 28, 29, 30]. Recently, however, there has been renewed interest [15, 17, 16, 18] (see Section 1.1), although as we are going see, Ruitenburg's discovery appears to deserve still broader attention.

Consider a propositional formula A. Fix a propositional variable p, which can be thought of as representing the context hole or the argument of A taken as a polynomial (other propositional variables being additional constants). Given any other formula B, write $A(B)$ for the result of substituting B for p. Also, write $A \equiv_L B$ for $\vdash_L A \leftrightarrow B$, where L is a chosen system of propositional logic. Now define the natural iterated substitution operation

$$A^0(p) := p, \quad A^{n+1}(p) := A(A^n(p)).$$

Such a sequence turns almost immediately into a cycle modulo $\equiv_{\mathsf{CPC}}$, i.e., the equivalence relation of Classical Propositional Calculus:

Lemma 1 ([39], Lemma 1.1)**.** *For any A,* $\quad A(p) \equiv_{\mathsf{CPC}} A^3(p)$.

The above observation can be reformulated as asserting that CPC has *uniformly globally periodic sequences* (ugps). A logic L has this property if there exist b, $c > 0$ s.t. for any formula A, $A^b(p) \equiv_L A^{b+c}(p)$. However, ugps has still a rather strong logical form: two existential quantifiers preceding an universal one. Hence one can consider changing the order of quantifiers to weaken the property:

(eventually) periodic sequences:

	globally			locally		
uniformly	$\exists b.$	$\exists c > 0.$	$\forall A.$	$\exists c > 0.$	$\forall A.$	$\exists b.$
parametrically	$\exists b.$	$\forall A.$	$\exists c > 0.$	$\forall A.$	$\exists b.$	$\exists c > 0.$

$A^b(p) \equiv_L A^{b+c}(p)$

*The results were obtained while the author was employed at FAU Erlangen-Nuremberg. In the final stages of preparing the print version, the author has been employed at University of Naples Federico II, supported by the PNRR MUR projects FAIR (No. PE0000013-FAIR).

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 41–57, doi:10.4204/EPTCS.435.4

So, do standard non-classical propositional calculi, IPC in particular, have at least *plps* (*parametrically locally periodic sequences*)?[1] If not, is there something special about CPC which makes results such as Lemma 1 possible?

One of many peculiarities of CPC as seen from the general perspective of *abstract algebraic logic* [4, 12] is that it is *finite*, i.e., determined by a single finite algebra of truth values. There are some other natural examples of finite logics (mostly some fuzzy logics of finite chains, such as Łukasiewicz three-valued logic), but in general, this property is rather rare among logics of importance in today's Computer Science. A somewhat more general property is *local finiteness*: a logic is locally finite if given a finite set of propositional variables, one can form only finitely many non-equivalent formulas. The modal system S5 is a typical example of a logic which is locally finite without being finite.

Lemma 2. *Any locally finite logic has plps.*

Sketch. Any sequence $p, A(p), A^2(p), A^3(p)\ldots$ disproving the plps property would also disprove local finiteness. □

It is, however, well-known that IPC is not locally finite: even in one propositional variable, there are infinitely many nonequivalent formulas. The exact description of the infinite algebra of formulas in one propositional variable is provided by the Rieger-Nishimura Theorem (see [38, 32] and [6, Ch. 7] for references, also Section 5 herein). As we are going to see in Section 2, most "natural" propositional logics which fail to be locally finite, fail to have (even parametrically locally) periodic sequences. IPC turns out to fare better. One can indeed show that (uniformly or parametrically) globally periodic sequences would be too much to expect, at least when formulas are allowed to contain other variables than p itself [39, §2]. But we do have

Theorem 3 ([39], Theorem 1.9)**.** IPC *has the ulps property: for any A, there exists b s.t.* $A^b(p) \equiv_{\mathsf{IPC}} A^{b+2}(p)$. *Moreover, b is linear in the size of A.*

In fact, Ruitenburg's theorem is effective: the proof provides an algorithm to compute b in question, as discussed in Sections 5 and 6 below. Moreover, as the periodic sequence property (in all its incarnations) transfers from sublogics to extensions in the same signature (just like local finiteness and unlike uniform interpolation), we also get that all superintuitionistic logics (*si-logics*) have ulps. This shows that unlike local finiteness, ulps does *not* guarantee the finite model property (*fmp*), or even Kripke completeness.

On the other hand, there is an obvious connection with fixpoint definability[2]: If $A(p)$ is a monotone formula, then $\{A^n(p)\}_{n\in\omega}$ stabilizes when reaching a cycle. In particular, substituting $\bot$ for p in $A^b(p)$ produces the least fixpoint, while substituting $\top$ produces the greatest fixpoints. This is why Mardaev [26, 27, 28, 29, 30] quotes Ruitenburg when investigating the issue of fixpoint definability in non-classical logics. Periodic sequences, however, are by no means the only way of ensuring that a logic has definable fixpoints: there are examples of systems having the latter property without the former. In fact, a combination of Pitts' [34] uniform interpolation with what modal logicians would describe as a

[1] I am not aware whether terminology and distinctions used in the present paper have been systematically introduced before. The original work of Ruitenburg uses the term *finite order*.

[2] In fact, the present author got interested in Ruitenburg's result for similar reasons, in the context of ongoing joint work with Albert Visser on definability of fixpoint in intuitionistic modal logics involving (a strong version of) the Löb axiom. I would like to thank Albert Visser for attracting my attention to Ruitenburg's work, for his support and comments on early drafts. Thanks are also due to Wim Ruitenburg for providing his recollections of how the theorem was proved. Furthermore, George Metcalfe kindly corrected my misunderstandings concerning the status of RM ("R with Mingle") and provided me with several additional references. Finally, I would like to thank the referees of this and earlier incarnations of this paper, Alexis Saurin, Lutz Schröder and the participants of Oberseminar of our group at FAU for discussions and suggestions.

(definable) *master modality*[3] plus some trivial additional restrictions would be sufficient. Ghilardi et al. [15, 16] discuss further the issue of computing fixpoints and fixpoint definability, and compare the two approaches. It is worth mentioning here that Ruitenburg in the final part of his paper suggests a potential connection with uniform interpolation, despite preceding Pitts' [34] by several years.

Finally, as suggested by a referee, a clarification might be in order. The reader might be aware of results on definability of fixpoints in modal logics (classical or intuitionistic ones) containing some form of the Löb axiom. Nevertheless, as pointed out in Corollaries 5 and 6, such logics generally fail the plps property. Such definability results concern *guarded* or *modalized* fixpoints, i.e., those where the fixpoint variable occurs only in the scope of a modal operator. Van Benthem [1] and Visser [45] illustrate how to use definability of such fixpoints to derive definability of ordinary, monotone fixpoints: Precisely here one can use the periodic sequence property for the underlying modality-free (*extensional*) reduct of the logic in question.

The purpose of this note is twofold. In Section 2, I discuss the status of periodic sequences in other non-classical logics, illustrating just how special the situation of IPC is. Starting from Section 3, I present a mechanization of Ruitenburg's result in the Coq proof assistant.[4]

1.1 Related work

In 2015–2016, when the mechanization described herein was produced, Ruitenburg's original and poorly understood syntactic proof[5] was the only available one. This in fact motivated the present author to take on the challenge of mechanizing the proof, despite a relatively limited experience with proof assistants at the time. In the meantime, Ghilardi and Santocanale [17, 18] provided a semantic proof using the apparatus developed in the Ghilardi and Zawadowski monograph [14], involving games for bounded bisimulations and (pre)sheaves over the category of finite rooted posets with bounded morphisms. Nevertheless, as Ghilardi and Santocanale admit, their semantic proof does not provide tight bounds and computational information provided by Ruitenburg's proof, and extracted by the Coq mechanization described in this paper. Furthermore, the hope expressed in their final remark

> While we can expect that periodicity phenomena of substitutions do not arise for the basic modal logic K, they surely do for locally tabular [i.e., locally finite] modal logics. Considering also the numerous results on definability of fixpoints ... these phenomena are likely to appear in other subsystems of modal logics. As far as we know, investigation of periodicity phenomena in modal logics is a research direction which has not yet been explored and where the bounded bisimulation methods might prove their strength once more.

in the light of Section 2 herein requires qualification: outside of locally finite modal logics, there seems to be little hope and scope for periodicity. Open Problem 1 below isolates one potential intuitionistic modal logic for which a generalization of Ruitenburg result might be possible. In fact, the mechanization described here can be used in investigating the problem or (should the answer turns out to be positive) verifying a potential proof; cf. Remark 9 in Section 3.1 and Remark 10 in Section 4.

[3] This is only needed if the logic in question contains additional "modal" connectives or lacks some structural rules. For intutionistic propositional logic itself, the requirement of having a "master modality" (global deduction theorem, equationally definable principal congruences...) is trivially satisfied, just like for standard relevance logics. On the other hand, this criterion is not generally met by substructural logics such as those covered by Theorem 8. They generally fail to satisfy axioms ensuring EDPC [13, Theorem 3.55]. Same problems arise with non-transitive modalities, even in the classical unimodal setting.

[4] The content of both parts is based on the work done in years 2015–2017, which for various reasons remained unpublished and presented only in the form of a talk at TACL 2017.

[5] It is worth mentioning here that Wim Ruitenburg himself (p.c.) claims that his original proof was utilizing Kripke semantics. Difficulties in explaining it to his colleagues, in particular Albert Visser, and Visser's additional insights finally recorded as Lemma 1.7 in Ruitenburg's paper, convinced him to cast the argument into a purely syntactic setting, which at the time proved clear enough to both Ruitenburg and Visser.

Of all Coq formalizations of non-trivial results concerning various propositional calculi, the recent work of Férée and van Gool [11] is probably closest to our interests here. It deals with Pitt's syntactic proof of uniform interpolation for IPC, which as discussed above provides another route towards fixpoint definability, and it also allows extraction of executable code, actually computing propositional quantifiers[6]. As Pitts' proof was cast in the setting of the terminating sequent calculus G4ip [22, 9] (Ruitenburg, by contrast, works with a slightly idiosyncratic and purely Hilbert-style setting, as discussed in Section 3), Férée and van Gool [11] mechanizes some metatheory of that calculus, in particular admissibility of a restricted form of cut and other structural rules. The present mechanization simply assumes decidability of IPC and does not attempt to provide either a syntactic proof via cut elimination or a Kripke-based semantic one, although such developments are available elsewhere.

Finally, there is an entire recent body of work mechanizing in Coq G4ip-style calculi for several propositional logics (over classical and intuitionistic base) developed by Shillito and coauthors [41, 42, 21]. It would seem of interest to turn the present formalization into a part of a larger library, integrating the developments described above and possibly other scattered contributions, such as the Coq development supporting the discussion of modal negative translations in Litak et al. [25].

2 Periodic sequences in nonclassical logics

In order to understand properly the special status of Ruitenburg's result, let us compare the situation in IPC with that in other non-classical logics, e.g., substructural or modal ones. It turns out that in almost all standard cases, plps (and even more so, ulps) implies local finiteness; IPC seems rather exotic in having the first property without the second one.

2.1 Modal logics over CPC

For modal logics over the boolean propositional base, the reader can refer to, e.g., Chagrov and Zakharyaschev [6] for notation, syntax and semantics; one difference is that I am using here a superscript $\cdot^{\mathsf{cl}}$ to make the CPC propositional base clear. For transitive modal logics, having periodic sequences is indistinguishable from local finiteness, i.e., the converse of Lemma 2 holds:

Theorem 4. *A normal extension of* $\mathsf{K4}^{\mathsf{cl}}$ *has plps iff it is locally finite.*

Proof. It is known [6, Theorem 12.21] that a normal modal logic extending K4 is locally finite iff it is of *infinite depth*, i.e., admits Kripke frames of arbitrary finite depths. Consider $A_{\mathsf{K4}}(p) := q \vee \Box(q \to \Box p)$. A straightforward modification of the argument proving the above equivalence [6, Theorem 12.21] shows the failure of plps (in the proof, the valuation for q should be defined in the same way as the valuation for p): the sequence $\{A^n_{\mathsf{K4}}(p)\}_{n\in\omega}$ never stabilizes. □

Corollary 5. *All extensions of* K^{cl} *contained in either* $\mathsf{S4Grz.3}^{\mathsf{cl}}$ *(such as* $\mathsf{K4}^{\mathsf{cl}}$*,* $\mathsf{S4}^{\mathsf{cl}}$*,* T^{cl}*) or* $\mathsf{GL.3}^{\mathsf{cl}}$ *(in particular* $\mathsf{GL}^{\mathsf{cl}}$*) fail to have locally periodic sequences.*

For subsystems of $\mathsf{GL.3}^{\mathsf{cl}}$, this can be proved via a simpler alternative technique that remains useful when the propositional base is weakened to IPC; see Theorem 7.

Moreover, even without transitivity it does not appear easy to find examples of logics with plps which are not locally finite. Shapirovsky [40] has provided an example of a normal modal logic which

[6] To make the relationship even stronger, the apparatus developed in the Ghilardi and Zawadowski monograph [14] provides a model-theoretic proof of both Pitts' result and Ruitenburg's result. In fact, the starting point for that monograph was their earlier article [19], explicitly motivated as "language-free" or categorical analysis of uniform interpolation. Visser [44] follows a similar approach based on bounded bisimulations, cast in somewhat less categorical terms.

has finitely many formulas in one variable, but fails local finiteness. Unfortunately, the technique used in the proof of Theorem 4 can be also applied to his example with a minor modification: namely, use

$$A_{\mathsf{Sha}}(p) := q \vee \Box(q \to \Box(p \vee r)),$$

where r is to be evaluated as $\{\omega\}$ in Shapirovsky's frame.

2.2 Intuitionistic modal logics

The reader is referred to the extensive literature [43, 46, 47, 24] for basic information about intuitionistic modal logics. Just for clarity, we are only discussing here intuitionistic modal logics with a single modality $\Box$ (no $\Diamond$), which is reflected in the notation. Theorem 4 immediately extends to intuitionistic modal logics being counterparts of standard extensions of K^{cl} (see Simpson's [43] Requirement 3):

Corollary 6. *All extensions of* $\mathsf{K}^{\mathsf{int}}_{\Box}$ *contained in either* $\mathsf{S4Grz.3}^{\mathsf{cl}}$ (*such as* $\mathsf{K4}^{\mathsf{int}}_{\Box}$, $\mathsf{T}^{\mathsf{int}}_{\Box}$,$\mathsf{S4}^{\mathsf{int}}_{\Box}$, $\mathsf{S4Grz.3}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{S4Grz.3}^{\mathsf{int}}_{\Box}$) *or* $\mathsf{GL.3}^{\mathsf{cl}}$ (*in particular* $\mathsf{GL}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{GL.3}^{\mathsf{int}}_{\Box}$) *fail to have locally periodic sequences.*

Some intuitionistic modal logics of computational interest have "degenerate" classical counterparts (see [24] for a discussion) and hence Corollary 6 cannot be used to disprove that they have periodic sequences. This includes $\mathsf{S}^{\mathsf{int}}_{\Box} := \mathsf{K}^{\mathsf{int}}_{\Box} \oplus A \to \Box A$, i.e., the Curry-Howard logic of *applicative functors*, also known as *idioms* [31]. Its classical counterpart S^{cl} and all its two consistent proper extensions are finite logics enjoying ulps. In fact, S^{cl} has exactly two proper consistent extensions, one denoted as Triv and the other denoted as Ver [6]. In contrast, not only does $\mathsf{S}^{\mathsf{int}}_{\Box}$ have uncountably many propositional extensions, but the failure of plps remains a common phenomenon among them. To show this, one can use a proof technique applicable to (subsystems of) logics with Löb-style axioms, either intuitionistic or classical ones:

Theorem 7. *No sublogic of* $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$, *also denoted as* $\mathsf{KM}_{\mathsf{lin}}$ [7] *has parametrically locally periodic sequences; this in particular applies to* $\mathsf{SL.3}^{\mathsf{int}}_{\Box} := \mathsf{S}^{\mathsf{int}}_{\Box} \oplus \mathsf{GL.3}^{\mathsf{int}}_{\Box}$, $\mathsf{SL}^{\mathsf{int}}_{\Box} := \mathsf{S}^{\mathsf{int}}_{\Box} \oplus \mathsf{GL}^{\mathsf{int}}_{\Box}$ *or* $\mathsf{S}^{\mathsf{int}}_{\Box}$.

Proof. The logic $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$ (or $\mathsf{KM}_{\mathsf{lin}}$) is the logic of the Kripke frame where the modal and the intuitionistic order are, respectively, irreflexive and reflexive variant of the reverse order on natural numbers. Consider $A_{\mathsf{KM}}(p) := \Box p$ and the valuation sending p to $\emptyset$; the denotation of $A^n_{\mathsf{KM}}(p)$ is the set of natural numbers smaller or equal to n. Hence, the sequence $\{A^n_{\mathsf{KM}}(p)\}_{n\in\omega}$ never stabilizes and plps fails in every logic sound in this frame. □

To contrast this with Theorem 4, note that $\mathsf{KM.3}^{\mathsf{int}}_{\Box}$, the propositional fragment of the logic of the Mitchell-Bénabou logic of the *topos of trees* [3, 7, 24], is *prefinite* or *pretabular*: all its extensions are finite, each determined by a finite chain. Interestingly, neither the proof Theorem 4 nor the proof of Theorem 7 apply to the Propositional Lax Logic $\mathsf{PLL}^{\mathsf{int}}_{\Box}$ [10], i.e., the Curry-Howard counterpart to (the type system of) Moggi's monadic metalanguage [2].

Open Problem 1. Does $\mathsf{PLL}^{\mathsf{int}}_{\Box}$ have locally periodic sequences?

2.3 Substructural logics

Arguments analogous to those above establish that in the realm of substructural logics [13], plps as a rule coincides with local finiteness. Consider $A_{\otimes}(p) := p \cdot p$, where $\cdot$ is the substructural *fusion* connective [13, §2.1.2], also known by linear logicians as *tensor* or *multiplicative conjunction* $\otimes$, and in the realm of the Logic of Bunched Implications BI and separation logic as *spatial*, *separating* or *independent* conjunction $*$ [37].

Theorem 8. *The product logic* Π*, the infinite valued Łukasiewicz logic* $Ł_\infty$ *or the logic of the* heap model *of* BBI (*boolean logic of bunched implications* [5, 33, 35, 37])*fail to have plps. Consequently, the property fails in all their sublogics, including* $(\mathsf{In}-)\mathsf{FL}_{(\mathsf{ew})}$*, multiplicative-additive fragment of linear logic* MALL (*and its intuitionistic fragment* IMALL) *and fuzzy logics such as* BL *or* MTL.[7]

Proof. This is shown by evaluating the sequence $\{A^n_\otimes(p)\}_{n\in\omega}$ defined above in the heap model or the $[0,1]$-interval with corresponding ℓ-norms. □

Thus, in order to find a natural substructural logic L enjoying the plps without local finiteness, one should look at those where the sequence $\{A^n_\otimes(p)\}_{n\in\omega}$ stabilizes modulo $\equiv_L$. It can be naturally achieved by stipulating that fusion is idempotent (both square-increasing and square-decreasing). This, however, is a very restrictive condition. When L satisfies the weakening rule, it collapses fusion to ordinary additive conjunction $\wedge$ and substructural implication to Heyting implication. Idempotent systems where $\cdot$ does not entirely collapse to $\wedge$ are sometimes considered by relevance logicians, with perhaps the most famous example being RM ("R with Mingle") . However, this system has been long known to be locally finite anyway [8]. See more recent references [36, 20] on limits of local finiteness results for idempotent structural logics.

Open Problem 2. Are there natural non-Heyting examples of (idempotent? square-increasing?) substructural logics with the plps property failing local finiteness?

3 Coq Formalization: The Basics

The formalization is available as a git repository

https://git8.cs.fau.de/software/ruitenburg1984.

It consists of less than 4000 lines of Coq code, split into 6 files. The code allows working program extraction to OCaml or Haskell; it can be also used directly for computation using Coq's core functional programming language (Gallina). More on that can be found in Section 6.

Naturally, the formalization involves a *deep* rather than *shallow* embedding of IPC. The syntax of IPC is formalized from first principles. Also, all semantic discussion (i.e., any mention of Kripke models) from the original paper is omitted. The semantic counterexamples given by Ruitenburg are unproblematic and easy to understand. The important part is purely syntactic.

3.1 Setup and basic lemmas

The language of IPC is defined as usual:

```
Inductive form :=
  | var : nat → form
  | imp : form → form → form
  | and : form → form → form
  | or : form → form → form
  | tt : form
  | ff : form.

Notation "A '&' B " := (and A B) (at level 40, left associativity).
Notation "A '\v/' B " := (or A B) (at level 45, left associativity).
Notation "A '->' B" := (imp A B) (at level 49, right associativity).
```

[7] See Galatos et al. [13] for substructural systems mentioned in the statement of this theorem.

```
Definition p := var 0.
Definition q := var 1.
Definition r := var 2.
```

Variable p will be used as a distinguished variable of the formula: the input or the argument of a polynomial. It is convenient to make it the variable with index 0 (consider the use of `destruct` in proofs where the distinguished variable should get a special treatment). We also explicitly add equivalence:

```
Notation "A '«->» B" := ((A -» B) & (B -» A)) (at level 58).
```

Ruitenburg's paper uses a formulation of IPC in terms of syntactic consequence (turnstile) relation between (finite) sets of formulas and formulas themselves. This approach is natural from the point of view of abstract algebraic logic [4, 12]. It would be natural to replace this turnstile-Hilbert-style axiomatization by a Gentzen-style formalism, either sequent calculus or natural deduction. We will return to this point in Section 3.2. The chosen formalization of IPC, however, is convenient for our purposes and stays as close to the development in the original article [39] as possible (Ruitenburg, in fact, did not write the exact axiomatization he was using or give an explicit reference for it, but it is easy to reconstruct).

Rather unsurprisingly, in the Coq version of the axiomatization I replaced finite sets of formulas with finite lists. The standard Hilbert-style presentation of IPC can be found in numerous references. In my setup, it looks as follows:

```
Reserved Notation "G '|–' A" (at level 63).

Notation context := (list form).

Inductive hil : context → form → Prop :=
 | hilst : ∀ G A, In A G → G |- A
 | hilK : ∀ G A B, G |- A -» B -» A
 | hilS : ∀ G A B C, G |- (A -» B -» C) -» (A -» B) -» A -» C
 | hilMP : ∀ G A B, G |- (A -» B) → (G |- A) → (G |- B)
 | hilC1 : ∀ G A B, G |- (A -» B -» A & B)
 | hilC2 : ∀ G A B, G |- (A & B -» A)
 | hilC3 : ∀ G A B, G |- (A & B -» B)
 | hilA1 : ∀ G A B, G |- (A -» A \v/ B)
 | hilA2 : ∀ G A B, G |- (B -» A \v/ B)
 | hilA3 : ∀ G A B C, G |- (A -» C) -» (B -» C) -» (A \v/ B -» C)
 | hiltt : ∀ G, G |- tt
 | hilff : ∀ G A, G |- ff -» A
                        where "G '|–' A" := (hil G A).
```

A sequence of lemmas follows in `HilbertIPCsetup.v`, establishing basic properties of the turnstile relation. There are also some easy tactics for use in later proofs. They should be all rather self-explanatory. Again, in Ruitenburg's paper trivial lemmas of this kind are used tacitly or nearly tacitly. Several, though by no means all of the basic lemmas in this part were added to the `Hint` database and so were, e.g., constructors of hil. This has in some cases slowed down working of some tactics, in particular `eauto`, but I believe overall the database has not been unduly swollen and `eauto`, `auto` and their cousins remain useful.

One also needs a standard notion of substitution; as the focus is entirely on a propositional language with no notions of—and no problems of—binding, α-conversion etc., I decided to use a straightforward, own formalization, with a minimal number of tailored tactics to make the development smoother. Base substitutions are simply functions from variables to formulas; we can thus reduce them to functions from natural numbers to formulas. They are extended inductively to arbitrary formulas and then to arbitrary contexts:

```
Fixpoint sub (s: nat → form) (A : form) : form :=
  match A with
    | var i ⇒ s i
    | A -» B ⇒ (sub s A) -» (sub s B)
    | A & B ⇒ (sub s A) & (sub s B)
    | A \v/ B ⇒ (sub s A) \v/ (sub s B)
    | tt ⇒ tt
    | ff ⇒ ff
  end.
Fixpoint ssub (s : nat → form) (G: context) : context :=
  match G with
    | nil ⇒ nil
    | A :: G' ⇒ (sub s A) :: (ssub s G')
  end.
```

Typical substitutions arise inductively from a base substitution replacing a single chosen variable by a formula and leaving all other variables unchanged:

```
Definition s_n (n:nat) (A: form) : (nat → form) :=
  fun m ⇒ match (eq_nat_dec n m) with
                | left _ ⇒ A
                | right _ ⇒ (var m)
            end.
Definition s_p := s_n 0.
Notation "A '{' B '}/' n " := (sub (s_n n B) A) (at level 29).
Notation "A '{' B '}/p'" := (sub (s_p B) A) (at level 29).
Notation "'ssubp' B G" := (ssub (s_p B) G) (at level 29).
```

As only the more narrow substitution s_p for the chosen variable is needed on most occasions, it does pay off to state suitable lemmas in two versions, one for s_n and one for s_n, the former usually with postfix _gen. There are also several notions of freshness, for formulas and for lists, both as a predicate and as a boolean-valued recursive function (all distinguished by corresponding suffixes). Finally, we can finalize the notion of iterated substitution:

```
Fixpoint f_p (A : form) (n : nat) : form :=
  match n with
    | 0 ⇒ var 0
    | S n' ⇒ sub (s_p (f_p A n')) A
  end.
```

Remark 9. The definitions so far were fairly straightforward. There are, however, two basic lemmas that are worth singling out and contrasting.

```
Lemma hil_ded: ∀ G (A B : form), A :: G |- B → G |- A -» B.
Lemma ded_subst_gen : ∀ A G B C n, G |- B «-» C →
                                   G |- (sub (s_n n B) A) «-» (sub (s_n n C) A).
```

As a consequence of the last lemma we have

```
Lemma ded_subst : ∀ A G B C, G |- B «-» C →
                             G |- (sub (s_p B) A) «-» (sub (s_p C) A).
```

For most non-classical logics, it is by no means common to have both of these metatheorems at the same time. In the modal logic setting, for example, a turnstile relation enjoying this deduction theorem

(i.e., the one of the form $\Gamma \cup \{A\} \vdash B$ implies $\Gamma \vdash A \to B$) would be the one known as the *local consequence relation*. However, this notion of consequence does not enjoy the substitution metatheorem. The global consequence relation, which incorporates the Rule of Necessitation, satisfies in turn the latter metatheorem, but not the former one. Similar problems would arise in the realm of substructural logics. Yet both these metatheorems are heavily used in Ruitenburg's work (implicitly) and the present formalization (explicitly), which indicates some reasons why the rarity of periodic sequences property outside the realm of locally finite logics may be not a coincidence. As far as substructural logics are concerned (at least those not enjoying the weakening rule), other examples of incompatible metatheorems heavily used in the present development would include:

```
Lemma hil_weaken: ∀ G (A B : form), G |- A → B :: G |- A.
Lemma hil_weaken_gen : ∀ G G' A, G |- A → G' ++ G |- A.
Lemma hil_weaken_incl : ∀ G G' A, G |- A → incl G G' → G' |- A.
```

However, $\mathsf{PLL}^{\mathsf{int}}_{\Box}$ singled out in Open Problem 1 and more broadly all extensions of $\mathsf{S}^{\mathsf{int}}_{\Box}$provides an important example of a logic enjoying simultaneously all metatheorems listed in the present Remark. Just like for intuitionism itself, there is no gap between its local consequence relation and its global counterpart. See also Remark 10 below.

3.2 Decidability of the turnstile relation

Ruitenburg's proof of `Theorem rui_1_4` in several places does case analysis, splitting cases between $\Gamma \vdash A$ and $\Gamma \nvdash A$. While in classical metatheory this does not require further justification, constructively of course it amounts to decidability of the turnstile relation. Simpler syntactic notions such as equality between variables or between formulas are trivially decidable (and, as one may guess, useful in formal development):

```
Lemma dceq_v : ∀ n n0, {var n = var n0} + {var n ≠ var n0}.
Lemma dceq_f: ∀ (A B: form), {A = B} + {A ≠ B}.
```

But a constructive proof of decidability of turnstile relation would require incorporating an actual decidability proof for IPC. As the present development is purely syntactic (and extending it with some standard Kripke completeness proof for IPC would provide little insight into Ruitenburg's proof), the only route worth considering would be the one mentioned above: give up the Hilbert-style approach of the original paper [39] and use a cut-free sequent system together with an actual proof of cut elimination, then use it to prove decidability of turnstile. This is a viable idea for future development, in particular if extended with proof term assignment to extract the computational content of Ruitenburg's result, especially in the light of more recent work discussed in Section 1.1. Still, it seems orthogonal to the actual goal of verifying the original proof. The route taken in the present paper looks as follows:

```
Module DecidEquiv.

Require Import Coq.Logic.Classical_Prop.

Lemma decid_equiv : ∀ G A, (G |- A) ∨ ~(G |- A).
  intros. apply classic.
Qed.

End DecidEquiv.
```

That is, excluded middle was imported inside a module in order not to contaminate the rest of development (admittedly, these days such import strategy is not encouraged by Coq developers). The reader can verify that it is only used in two places in the proof of `Theorem rui_1_4`.

4 Proving Ruitenburg's auxiliary lemmas

So far, we were discussing basic metatheorems used implicitly in Ruitenburg's paper. In this section, we are finally going to start formalizing the actual development in the paper itself, corresponding Lemma 1.2 and Lemmas 1.6–1.8 in the original paper. Lemma 1.2 is the very last part of `HilbertIPCSetup`:

```
Lemma rui_1_2_i : ∀ A i k, [f_p A i ; sub (s_p tt) A] |- f_p A (i + k).
Lemma rui_1_2_ii : ∀ A i n, [ f_p A i; sub (s_p tt) A ] |-
                                (f_p A (S n) -» f_p A n) -» f_p A n.
```

Lemmas 1.6–1.8 form the entirety of `Ruitenburg1984Aux`:

```
Lemma rui_1_6 : ∀ A m,
                  (∀ i, [f_p A i ; sub (s_p tt) A] |- f_p A m) →
                                [sub (s_p tt) A] |- f_p A m «-» f_p A (m+1).
Lemma rui_1_7_i : ∀ A m n,
                   [ sub (s_p tt) (f_p A (2×m + 1))] |- sub (s_p tt) (f_p A n).
Lemma rui_1_7_ii : ∀ A m n,
                 [ sub (s_p tt) (f_p A (2×m + 2))] |- sub (s_p tt) (f_p A (2 ×n)).
Lemma rui_1_8 : ∀ A m,
                   [sub (s_p tt) A] |- (f_p A m) «-» (f_p A (m + 1)) →
                                [] |- f_p A (m+1) «-» f_p A (m + 3).
```

As one can see, these are rather nontrivial metatheorems about IPC. Still, it was a rather pleasant part of the paper to formalize. The automated proofs follow closely proofs in the paper. I would even claim that the Coq proofs are at times easier to understand than in the original version and clarify some remarks that were not always entirely transparent—e.g., the repeated instruction to "use (iterated) substitution"—but this is ultimately a question of individual preferences.

Remark 10. It is worth noting that unlike the main theorem itself (`Theorem rui_1_9_Ens`) and its key ingredient (`Theorem rui_1_4`) presented in Section 5 below, all the lemmas in question would hold for systems meeting the restrictions discussed in Remark 9, in particular to intuitionistic modalities satisfying the axiom $\mathsf{S}^{\mathsf{int}}_{\Box}$, i.e., Curry-Howard counterparts of applicative functors. In fact, the repository includes a fork `feature/extended_formalisms` with a subfolder `applicative_development` illustrating that all the results of `HilbertIPCSetup` and `Ruitenburg1984Aux` (i.e., the parts of formalization presented in Section 3.1 and in this section) would work for any extension $\mathsf{S}^{\mathsf{int}}_{\Box}$ in the unimodal Heyting signature. Whether or not there are interesting non-locally-finite logics of this form for which the rest of the development can be carried, i.e., for which llps holds is not clear; cf. Open Problem 1.

5 Bounds as Ensembles: proving the main result

`Theorem rui_1_4` relies on a notion of a *bound* of formula A over a context (a set, or in our case a list of formulas). It is a finite set of formulas, each of which is equivalent to a substituted implicational subformula of A (more precise definition below), and the proof of `Theorem rui_1_4` proceeds by induction over its cardinality. Ensemble, an old weapon in Coq's arsenal, seems particularly well-suited for such proofs. The disadvantage, of course, is that being a predicate, i.e., a `Prop`-valued function, it does not allow the use of program extraction or Coq's programming capabilities; we will see a solution in Section 6. I believe, however, that it was beneficial to keep the logical and computational uses of bounds apart. If the code is refined in future, it could be beneficial to explicitly use the notion of *reflection* here.

After giving the obvious definition of Subformulas, we identify their special subclass BoundSubformulas: those which are either implicational subformulas or propositional variables. Then one can proceed to defining what bounds are:

```
Definition Bound (G: context) (A : form) (b : Ensemble form) :=
  ∀ C: form, In (BoundSubformulas A) C →
                  ∃ B , In b B ∧ G |- sub (s_p tt) C «-» B.
```

There is a minor discrepancy between the definition of Bound used here and the one used in the original paper [39]: the latter does not impose that b already contains formulas (equivalent to) BoundSubformulas of A elements p substituted with tt; in Ruitenburg's version, the part following the turnstile would be sub (s_p tt) C «-» (s_p tt) B. Of course, if b is a bound in his sense, then $b\{tt/p\}$ is a bound both in the present sense and in his sense; hence, the present definition is narrower, but the difference does not matter from a practical point of view. On the other hand, Ruitenburg's definition insist that a bound contains tt, whereas it may well happen that a Bound contains nothing equivalent to it; consider, for example, a bound of q -» r over []. The difference can be handled easily, as we will see in the statement of Theorem rui_1_4 below; moreover, Ruitenburg's convention is followed when bounds are treated as lists rather than Ensembles (see Section 6).

```
Definition ExactBound (G: context) (A : form) (b : Ensemble form) :=
  (∀ B: form, In b B →
                  ∃ C , In (BoundSubformulas A) C ∧
                      G |- sub (s_p tt) C «-» B) ∧
  Bound G A b
```

The remainder of BoundsSubformulas.v is devoted to various auxiliary lemmas.

We can move on now to the contents of Ruitenburg1984KeyTheorem, which contains the actual proof of the central syntactic result in the paper. The first larger theorem proved in this file, before actual Theorem rui_1_4, deals with a remark in the base case of the proof, referring to the Rieger-Nishimura theorem mentioned in Section 1. Recall that this very theorem pinpoints why IPC does not have local finiteness by describing the infinite poset of all IPC-formulas in one free variable, quotiented by provable equivalence and ordered by provable implication. While the Rieger-Nishimura lattice has been thoroughly understood and reconstructed on several occasions, using differing techniques[8], it could be of interest to formalize it fully. Fortunately, as it turns out, we only need a corollary of this result:

```
Lemma Rieger_Nishimura_corollary :∀ G B v,
            fresh_l_p v (B :: G) →
            ( ∀ C : form, (BoundSubformulas B) C → G |- sub (s_p tt) C ) → ∀ C, (Subformulas B)
C → G |- (sub (s_p (var v)) C) -» ff ∨
                                  G |- sub (s_p (var v)) C «-» (var v) ∨
                                  G |- sub (s_p (var v)) C.
```

With this last issue out of the way, one can finally state and prove

```
Theorem rui_1_4: ∀ n G A B,
  Included (BoundSubformulas B) (BoundSubformulas A) →
  ∀ i v,
  let G' := (f_p A i) :: sub (s_p tt) A :: G in
  (∃ b, let b' := (App tt b) in Bound G' A b' ∧ cardinal b n) →
  fresh_l_p v (p::B::A::G) →
```

[8] In fact, its very name refers to the rediscovery of Rieger's result [38] by Nishimura [32]. More information and further references can be found in standard monographs (see, e.g., [6, Ch. 7]).

```
((f_p A (2×n)) -» (var v)) :: G' |-
  (sub (s_p (var v)) B «-» sub (s_p tt) B) & (sub (s_p tt) B -» (var v)) ∨
((f_p A (2×n)) -» (var v)) :: G' |- sub (s_p (var v)) B «-» (var v) ∨
((f_p A (2×n)) -» (var v)) :: G' |- sub (s_p (var v)) B.
```

A comparison reveals inessential differences with the statement of this theorem in the original paper. Instead of assuming that BoundSubformulas of B are contained in those of A, a premise of the result stated by Ruitenburg is the existence of a bound (of size n) of $A \,\&\, B$ over G'. Then, at the very beginning of the proof, an observation is made that one can assume that *B is a subformula of A by replacing A by the equivalent formula* $A \,\&\, (B \setminus \text{v/ tt})$. The assumption made here, i.e., that (BoundSubformulas B) are included in (BoundSubformulas A) seems an optimal solution. Another, very minor difference is that the supposed bound b is immediately tweaked to b' containing tt. This has to do with the difference in the definition of our Bound mentioned above.

The proof is by induction on n, i.e., the cardinality of a bound for A (and, consequently, also for B). The inductive step, furthermore, involves an induction over B. This in turn involves a consideration of numerous cases and subcases of these subcases, almost each of which involves some actual propositional reasoning in IPC.

The theorem yields corollaries grouped in the file `Ruitenburg1984Main.v`:

```
Corollary rui_1_4': ∀ A b n, (Bound [] A b) → cardinal b n →
  ∀ i v, let v' := S v in fresh_f_p v' A →
     let G' := [f_p A i ; sub (s_p tt) A] in
     let G'' := (f_p A (2×n) -» (var v')) :: G' in
       G'' |- (sub (s_p (var v')) A «-» sub (s_p tt) A) &
              (sub (s_p tt) A -» (var v')) ∨
       G'' |- sub (s_p (var v')) A «-» (var v') ∨
       G'' |- sub (s_p (var v')) A.
```

In other words, this corollary is simply stating what rui_1_4 means for a single formula A rather than for a pair of formulas A, B (note one needs to ensure here that the fresh variable in question is not p itself; it might happen that A does not contain it).

```
Corollary rui_1_5: ∀ A b i n, (Bound [] A b) → cardinal b n →
     let m:= (2 × n + 1) in
       [sub (s_p tt) A; f_p A i] |- f_p A m.
```

While it may seem surprising, proving this much simpler-looking corollary is the only goal of the intimidating `Theorem` rui_1_4, and the proof of this corollary is not even using the theorem itself, but rather `Corollary` rui_1_4' above. Note also that in the original paper, the statement of the theorem does not involve the connection between m and a bound, although it is clear in the proof.

This corollary is now combined with Lemmas 1.6–1.8 discussed in Section 4 to yield

```
Theorem rui_1_9_Ens:
  ∀ A b m, (Bound [] A b) →
                  cardinal b m →
                  [] |- f_p A (2 × m + 2) «-» f_p A (2 × m + 4).
```

This finally explains the name Bound: the size of such a bound for A determines (linearly) how many iterated substitutions (at worst) it takes before it enters the cycle. Clearly, there is always a bound linear in the size of A: simply take all the implicational subformulas (i.e., BoundSubformulas) of A and substitute tt for p removing possible duplicates. Is it the best that one can do? And is it important to care about such minor adjustments? This is the last part of our considerations.

6 Bounds as lists: computations and program extraction

As discussed in Section 5, treatment of bounds as Ensembles, very convenient for the proof of main result, is not very useful computationally, starting from the fact that `Prop` gets erased during program extraction. For this reason, one can consider a natural reformulation of the notion of a bound in terms of lists. These, in turn, would be awkward in proofs discussed in Section 5, but are bread-and-butter from a functional programming point of view. All that is needed to verify the programs obtained in this way is to provide bridge theorems between Ensemble- and list-counterparts of the same notion, and this is much easier than developing everything from scratch in terms of lists. These are the contents of `BoundsLists.v`. As suggested above, a more structured approach would probably involve systematic use of reflection.

A suitable counterpart of Bound is

```
Definition bound (b: list form) (A : form) (G : context) :=
  Forall (fun C ⇒ Exists (fun B ⇒ G |- B «-» C ∧ [] |- B «-» sub (s_p tt) B) b) (mb_red A).
```

Using a natural function converting contexts (i.e., lists) to Ensembles, one can easily show

```
Lemma bound_is_Bound : ∀ b A G, bound b A G → Bound G A (context_to_set b).
```

And the corresponding version of the main theorem is

```
Theorem rui_1_9_list: ∀ A b, (bound b A []) →
  ∃ m, m ≤ length b ∧[] |- f_p A (2 × m + 2) «-» f_p A (2 × m + 4).
```

The most naïve way to produce a bound for A over `[]` (and hence over any G, as implied by bound_for_bound_upward) is by

```
Fixpoint mb_red (A : form) : list form :=
match A with
    | var i ⇒ [sub (s_p tt) (var i) ; tt]
    | B -» C ⇒ sub (s_p tt) (B -» C) :: (mb_red B ++ mb_red C)
    | B & C ⇒ (mb_red B ++ mb_red C)
    | B \v/ C ⇒ (mb_red B ++ mb_red C)
    | tt ⇒ [tt]
    | ff ⇒ [tt]
end.
```

Note that this time we are following Ruitenburg's convention and explicitly including `tt`.

However, this is obviously suboptimal. To begin with, the output of mb_red is almost guaranteed to contain duplicates, but this is easy to deal with (using dup_rem). More importantly, such a list is also likely to contain equivalent formulas, which are also redundant. The problem gets dramatic when the formula in question contains no other variables than p; cf. Proposition 2.3 and Theorem 2.4 in the original paper [39]; within the one-variable fragment, IPC has strictly globally periodic sequences, just like the classical logic. But improvements are possible also when a formula contains more than one variable. The present development is restricted to a simple optimizer t_optimize, which is essentially removing redundant occurrences of tt. The function optimized_bound combines duplicate removal from dup_rem with iterating t_optimize as many times as the formula depth of A requires. Still further improvements are conceivable. Given that IPC is decidable, the ultimate option would be to integrate a decision procedure for IPC (cf. Section 3.2) and test pairwise elements of a given bound, removing the elements equivalent to those found earlier in the list.

Nevertheless, the present stage of development can already be used for actual computation, either via Coq's core functional programming language Gallina or, if one prefers, via program extraction. Combo functions available at the end of `Ruitenburg1984Main.v`, i.e., *optimized_cycle* or *cycle_formula_length* produce the value after which the sequence is going to enter a cycle for a given $A(p)$ and the size of corresponding $A^{2m+2}(p)$. They can be directly extracted to any typical target language such as Haskell or OCaml. In fact, Coq's `Compute` itself does a satisfying job in computing these values. However, even simple experiments indicate one should be rather careful as a blow-up can occur very quickly. The fact that m itself is linear in the size of A surely enough does not mean that $A^{2m+2}(p)$ is and rather simple examples can make it painfully clear. Consider, e.g., a formula from `BoundsLists.v`:

```
Definition exform1 := (q -» p -»r) & ((p -» r) -» p \v/ r).
```

for which the length of the value of optimized_boundoptimized_bound is just $m := 4$. The reader is now encouraged to pick $A(p)$ to be exform1 and, as an exercise, estimate the size of $A^{2m+2}(p)$.[9]

7 Conclusions

As the formalization was developed in 2015–2016, and it was an exercise for the author in understanding Ruitenburg's paper and improving his own skills, it does not involve most modern or complex Coq libraries and features. After relatively minor changes, it has proved possible to compile under recent versions of Coq (upwards of 8.17), but the development itself is not using in an essential way anything that was not already available in versions 8.4pl6 and 8.5. Some libraries being used are already getting obsolete, but a proper overhaul would constitute a separate project, focusing directly on the theorem-proving community. Section 1.1 and the work of Férée and van Gool [11] or Shillito and coauthors [41, 42, 21] suggest how such an overhaul could potentially look like.

The routes for future development have been already suggested in the paper. I find the question whether there are other natural non-locally-finite logics with llps particularly intriguing (Open Problems 1 and 2). Combining the present formalization with some standard proof of decidability of IPC and using it, e.g., to eliminate altogether the meta-level Excluded Middle (Section 3.2) or to compute optimal size of a bound for any input (Section 6) also seems a natural challenge for future work.

References

[1] Johan van Benthem (2006): *Modal Frame Correspondences and Fixed-Points*. *Studia Logica* 83(1-3), pp. 133–155, doi:10.1007/s11225-006-8301-9.

[2] P. N. Benton, Gavin M. Bierman & Valeria de Paiva (1998): *Computational Types from a Logical Perspective*. *J. Funct. Program.* 8(2), pp. 177–193, doi:10.1017/S0956796898002998.

[3] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer & Kristian Støvring (2012): *First Steps in Synthetic Guarded Domain Theory: Step-Indexing in the Topos of Trees*. *LMCS* 8, pp. 1–45, doi:10.2168/LMCS-8(4:1)2012.

[4] W. J. Blok & D. Pigozzi (1989): *Algebraizable logics*. *Memoirs AMS* 77 (396), AMS.

[5] James Brotherston & Max Kanovich (2014-04): *Undecidability of Propositional Separation Logic and Its Neighbours*. *J. ACM* 61(2), pp. 14:1–14:43, doi:10.1145/2542667.

[6] A. Chagrov & M. Zakharyaschev (1997): *Modal Logic*. *Oxford Logic Guides* 35, Clarendon Press.

[9] This is, in fact, a mistake I made myself while experimenting with the code. I ran Gallina on this input without a prior pen-and-paper or simply commonsensical estimate of the size of output. To Gallina's credit, it was able to return with the actual formula after several minutes. A curious reader can see it in `f_p_exform1_10_output.txt`.

[7] Ranald Clouston & Rajeev Goré (2015): *Sequent Calculus in the Topos of Trees*. In Andrew M. Pitts, editor: *Proceedings of FoSSaCS 2015*, *LNCS* 9034, Springer, pp. 133–147, doi:10.1007/978-3-662-46678-0_9.

[8] J. Michael Dunn (1970): *Algebraic Completeness Results for R-Mingle and Its Extensions*. *The Journal of Symbolic Logic* 35(1), pp. 1–13, doi:10.2307/2271149.

[9] Roy Dyckhoff (1992): *Contraction-free sequent calculi for intuitionistic logic*. *Journal of Symbolic Logic* 57, pp. 795–807, doi:10.2307/2275431.

[10] Matt Fairtlough & Michael Mendler (1997): *Propositional Lax Logic*. *Information and Computation* 137(1), pp. 1–33, doi:10.1006/inco.1997.2627.

[11] Hugo Férée & Sam van Gool (2023): *Formalizing and Computing Propositional Quantifiers*. In Robbert Krebbers, Dmitriy Traytel, Brigitte Pientka & Steve Zdancewic, editors: *Proceedings of CPP 2023*, ACM, pp. 148–158, doi:10.1145/3573105.3575668.

[12] Josep Maria Font, Ramon Jansana & Don Pigozzi (2003): *A Survey of Abstract Algebraic Logic*. *Studia Logica* 74(1-2), pp. 13–97, doi:10.1023/A:1024621922509.

[13] Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski & Hiroakira Ono (2007): *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. *Studies in Logic and the Foundations of Mathematics* 151, Elsevier.

[14] S. Ghilardi & M. Zawadowski (2002): *Sheaves, Games and Model Completions*. *Trends In Logic, Studia Logica Library* 14, Kluwer, doi:10.1007/978-94-015-9936-8.

[15] Silvio Ghilardi, Maria João Gouveia & Luigi Santocanale (2016): *Fixed-Point Elimination in the Intuitionistic Propositional Calculus*. In Bart Jacobs & Christof Löding, editors: *Proceedings of FoSSaCS 2016*, Springer Berlin Heidelberg, pp. 126–141, doi:10.1007/978-3-662-49630-5_8.

[16] Silvio Ghilardi, Maria João Gouveia & Luigi Santocanale (2020): *Fixed-point Elimination in the Intuitionistic Propositional Calculus*. *ACM Trans. Comput. Log.* 21(1), pp. 4:1–4:37, doi:10.1145/3359669.

[17] Silvio Ghilardi & Luigi Santocanale (2018): *Ruitenburg's Theorem via Duality and Bounded Bisimulations*. In Guram Bezhanishvili, Giovanna D'Agostino, George Metcalfe & Thomas Studer, editors: *Proceedings of AiML 2018*, College Publications, pp. 277–290. Available at `http://www.aiml.net/volumes/volume12/Ghilardi-Santocanale.pdf`.

[18] Silvio Ghilardi & Luigi Santocanale (2020): *Free Heyting algebra endomorphisms: Ruitenburg's Theorem and beyond*. *Math. Struct. Comput. Sci.* 30(6), pp. 572–596, doi:10.1017/S0960129519000203.

[19] Silvio Ghilardi & Marek Zawadowski (1995): *A sheaf representation and duality for finitely presented Heyting algebras*. *Journal of Symbolic Logic* 60, pp. 911–939, doi:10.2307/2275765.

[20] José Gil-Férez, Peter Jipsen & George Metcalfe (2020): *Structure theorems for idempotent residuated lattices*. *Algebra universalis* 81(2), p. 28, doi:10.1007/s00012-020-00659-5.

[21] Rajeev Goré, Revantha Ramanayake & Ian Shillito (2021): *Cut-Elimination for Provability Logic by Terminating Proof-Search: Formalised and Deconstructed Using Coq*. In Anupam Das & Sara Negri, editors: *Proceedings of TABLEAUX 2021*, *Lecture Notes in Computer Science* 12842, Springer, pp. 299–313, doi:10.1007/978-3-030-86059-2_18.

[22] Jörg Hudelmaier (1989): *Bounds for cut elimination in intuitionistic propositional logic*. Ph.D. Thesis, University of Tübingen.

[23] Lloyd Humberstone (2011): *The Connectives*. MIT Press, doi:10.7551/mitpress/9055.001.0001.

[24] Tadeusz Litak (2014): *Constructive Modalities with Provability Smack*. In Guram Bezhanishvili, editor: *Leo Esakia on Duality in Modal and Intuitionistic Logics*, Springer Netherlands, Dordrecht, pp. 187–216, doi:10.1007/978-94-017-8860-1_8.

[25] Tadeusz Litak, Miriam Polzer & Ulrich Rabenstein (2017): *Negative Translations and Normal Modality*. In Dale Miller, editor: *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 84, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 27:1–27:18, doi:10.4230/LIPIcs.FSCD.2017.27.

[26] Sergey I. Mardaev (1993): *Least fixed points in Grzegorczyk's logic and in the intuitionistic propositional logic*. *Algebra and Logic* 32(5), pp. 279–288, doi:10.1007/BF02261708.

[27] Sergey I. Mardaev (1994): *Convergence of positive schemes in S4 and Int*. *Algebra and Logic* 33(2), pp. 95–101, doi:10.1007/BF00739995.

[28] Sergey I. Mardaev (1997): *Fixed points of modal negative operators*. *Bull. Sect. Log., Univ. Lodz, Dep. Log* 26, pp. 135–138.

[29] Sergey I. Mardaev (1998): *Negative modal schemes*. *Algebra and Logic* 37(3), pp. 187–191, doi:10.1007/BF02671590.

[30] Sergey I. Mardaev (2007): *Definable fixed points in modal and temporal logics—a survey*. *Journal of Applied Non-Classical Logics* 17(3), pp. 317–346, doi:10.3166/jancl.17.317-346.

[31] Conor McBride & Ross Paterson (2008): *Applicative programming with effects*. *J. Funct. Program.* 18(1), pp. 1–13, doi:10.1017/S095679680700632.

[32] Iwao Nishimura (1960): *On Formulas of One Variable in Intuitionistic Propositional Calculus*. *The Journal of Symbolic Logic* 25(4), pp. 327–331, doi:10.2307/2963526.

[33] Peter W. O'Hearn & David J. Pym (1999): *The Logic of Bunched Implications*. *The Bulletin of Symbolic Logic* 5(2), pp. 215–244, doi:10.2307/421090.

[34] Andrew M. Pitts (1992): *On an interpretation of second order quantification in first order intuitionistic propositional logic*. *The Journal of Symbolic Logic* 57, pp. 33–52, doi:10.2307/2275175.

[35] David J. Pym, Peter W. O'Hearn & Hongseok Yang (2004): *Possible worlds and resources: the semantics of BI*. *Theoretical Computer Science* 315(1), pp. 257–305, doi:10.1016/j.tcs.2003.11.020. Mathematical Foundations of Programming Semantics.

[36] James Raftery (2007): *Representable idempotent commutative residuated lattices*. *Trans. Am. Math. Soc.* 359, doi:10.1090/S0002-9947-07-04235-3.

[37] John C. Reynolds (2002): *Separation Logic: A Logic for Shared Mutable Data Structures*. In: *Proceedings of LiCS 2002*, IEEE Computer Society, pp. 55–74, doi:10.1109/LICS.2002.1029817.

[38] Ladislav Rieger (1949): *On the lattice theory of Brouwerian propositional logic*. *Acta Facultatis Rerum Naturalium Universitatis Carolinae* 189, F. Řivnáč, Prague, doi:10.2307/2266273.

[39] W. Ruitenburg (1984): *On the period of sequences $(A^n(p))$ in intuitionistic propositional calculus*. *Journal of Symbolic Logic* 49, pp. 892–899, doi:10.2307/2274142.

[40] Ilya B. Shapirovsky (2021): *Glivenko's Theorem, Finite Height, and Local Tabularity*. *FLAP* 8(8), pp. 2333–2348. Available at https://collegepublications.co.uk/ifcolog/?00050.

[41] Ian Shillito, Iris van der Giessen, Rajeev Goré & Rosalie Iemhoff (2023): *A New Calculus for Intuitionistic Strong Löb Logic: Strong Termination and Cut-Elimination, Formalised*. In Revantha Ramanayake & Josef Urban, editors: *Proceedings of TABLEAUX 2023*, *Lecture Notes in Computer Science* 14278, Springer, pp. 73–93, doi:10.1007/978-3-031-43513-3_5.

[42] Ian Shillito & Rajeev Goré (2022): *Direct elimination of additive-cuts in GL4ip: verified and extracted*. In David Fernández-Duque, Alessandra Palmigiano & Sophie Pinchinat, editors: *Proceedings of AiML 2022*, College Publications, pp. 429–450.

[43] Alex K. Simpson (1994): *The Proof Theory and Semantics of Intuitionistic Modal Logic*. phdthesis, University of Edinburgh. https://era.ed.ac.uk/handle/1842/407.

[44] Albert Visser (1996): *Uniform Interpolation and Layered Bisimulation*. In P. Hájek, editor: *Gödel '96, Logical Foundations of Mathematics, Computer Science and Physics —Kurt Gödel's Legacy*, Springer, pp. 139–164. Reprinted as Lecture Notes in Logic 6, Association of Symbolic Logic.

[45] Albert Visser (2005): *Löb's Logic Meets the μ-calculus*. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk & Roel C. de Vrijer, editors: *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, *Lecture Notes in Computer Science* 3838, Springer, pp. 14–25, doi:10.1007/11601548_3.

[46] Frank Wolter & Michael Zakharyaschev (1997): *On the relation between intuitionistic and classical modal logics*. *Algebra and Logic* 36, pp. 121–125, doi:10.1007/BF02672476.

[47] Frank Wolter & Michael Zakharyaschev (1998): *Intuitionistic Modal Logics as fragments of Classical Modal Logics*. In Ewa Orlowska, editor: *Logic at Work, Essays in honour of Helena Rasiowa*, Springer–Verlag, pp. 168–186.

Nominal Algebraic-Coalgebraic Data Types, with Applications to Infinitary λ-Calculi

A short* fanfiction on [19]

Rémy Cerda†
Aix-Marseille Univ., CNRS, I2M
rcerda@math.cnrs.fr

Ten years ago, it was shown that nominal techniques can be used to design coalgebraic data types with variable binding, so that α-equivalence classes of infinitary terms are directly endowed with a corecursion principle [19]. We introduce "mixed" binding signatures, as well as the corresponding type of mixed inductive-coinductive terms. We extend the aforementioned work to this setting. In particular, this allows for a nominal description of the sets Λ^{abc} of abc-infinitary λ-terms (for $a, b, c \in \{0, 1\}$) and of capture-avoiding substitution on α-equivalence classes of such terms.

α-equivalence, the relation on λ-terms obtained by renaming bound variables, is central in λ-calculus (as in any syntax with binding): it is crucially needed in order to define capture-avoiding substitution in a satisfactory (*i.e.* total) manner, and thus to define β-reduction. Even though there are several well-known treatments of it — *via* the classical "variable convention" [6], or using "de Bruijn indices" [9] more suited to computer-assisted formalisations — giving abstract and canonical presentations of the operations of quotient by α-equivalence and capture-avoiding substitution has been pursued by several lines of research in the last decades. Such presentations have been proposed *via* the introduction of binding algebras [13], nominal sets [14, 26] or more recently De Bruijn algebras [15].

In infinitary λ-calculi [16, 8], the precise definition of α-equivalence is not as standard and straightforward as in a finite setting, in particular because some issues arise from the possibility to encounter terms containing free occurrences of *all* the available variables. Applying nominal techniques to the study of infinitary terms led Kurz, Petrişan, Severi, and de Vries to establish a canonical, abstract framework for defining α-equivalence in a coalgebraic setting [18, 19].

They conclude their work by suggesting that this framework could be applied not only to the "full" infinitary λ-calculus Λ^{111}, but also to its "mixed" inductive-coinductive variants, *e.g.* Λ^{001} [16, 12]. Doing so is the point of this small fanfiction[1]. Our contribution is twofold:

1. We provide an adapted framework for general "mixed" terms with binding by introducing *mixed binding signatures* (MBS). The main difference in their categorical treatment is that we replace 1-variable polynomial functors with 2-variable ones (*i.e.* bifunctors).
2. We show that the proof of [19] can be easily adapted to this slightly more general setting. As an example, we define capture-avoiding substitution on Λ^{001} by mixed recursion and corecursion.

*A long version of this abstract can be downloaded from the author's webpage, and appeared as a part of [10].

†The author wishes to thank Dimitri Ara, whose categorical knowledge was a great help, as well as Léo Hubert, Étienne Miquey and Lionel Vaux Auclair for helpful and stimulating discussions, and an anonymous referee who suggested several highly pertinent references.

[1]By using that word, we want to make clear that we do not claim much originality in the leading ideas of this work; we follow the very same path as [19], and we perform the necessary adaptions to lift their results to an inductive-coinductive setting.

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 59–70, doi:10.4204/EPTCS.435.5

1 Mixed binding signatures and mixed terms

In this section, we introduce *mixed* binding signatures as well as the finite and infinitary terms arising from such a signature. Then we extend to this setting all the metric and nominal structures one considers when dealing with ordinary binding signatures, and we describe a problem similar to what [19] solves in the ordinary setting.

1.1 Nominal preliminaries

Let us first recall a few basic definitions and properties about nominal sets. We remain quite superficial, since most of the nominal machinery is hidden in this paper; we refer to the excellent summary in [19, Sec. 4], from which we take all our notations, and to the standard literature on the subject [26].

Fix a set $\mathcal{V}$ of *variables*[2] and denote by $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$ the group of the permutations of $\mathcal{V}$ that are generated by transpositions $(x\ x')$, *i.e.* such that $\{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ is finite. A nominal set $(A, \cdot)$ is a set A equipped with a $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-action $\cdot$ such that each $a \in A$ is *finitely supported*, *i.e.* there exists a least finite set $\mathrm{supp}(a) \subset \mathcal{V}$ such that

$$\forall \sigma \in \mathfrak{S}_{\mathsf{fs}}(\mathcal{V}),\ (\forall x \in \mathrm{supp}(a),\ \sigma(x) = x) \Rightarrow \sigma \cdot a = a.$$

Intuitively, variables in $\mathrm{supp}(a)$ are "free in a". Nominal sets together with $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-equivariant maps form a category **Nom**.

The key object in all what follows is the *abstraction functor* $[\mathcal{V}]- : \mathbf{Nom} \to \mathbf{Nom}$ defined as follows. Fix a nominal set $(A, \cdot)$. $\mathcal{V} \times A$ is equipped with an equivalence relation $\sim$ defined by

$$(x, a) \sim (x', a') \quad \text{whenever} \quad \exists y \notin \mathrm{supp}(a) \cup \mathrm{supp}(a') \cup \{x, x'\},\ (x\ y) \cdot a = (x'\ y) \cdot a'.$$

The intuition behind $\sim$ is that it equates elements of A modulo renaming of free occurrences of a single given variable. We denote by $\langle x \rangle a$ the class of (x, a) in $(\mathcal{V} \times A)/\!\sim$, and we define a $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-action on such classes by $\sigma \cdot \langle x \rangle a \coloneqq \langle \sigma(x) \rangle (\sigma \cdot a)$. The functor $[\mathcal{V}]-$ is defined by $[\mathcal{V}]A \coloneqq (\mathcal{V} \times A)/\!\sim$ on objects, and $[\mathcal{V}]f : \langle x \rangle a \mapsto \langle x \rangle f(a)$ on morphisms.

The reverse construction is *concretion*, *i.e.* the partial equivariant map $[\mathcal{V}]A \times \mathcal{V} \to A$ defined by $(\langle x \rangle a, y) \mapsto \langle x \rangle a \mathbin{@} y \coloneqq (x\ y) \cdot a$ for $y \notin \mathrm{supp}(\langle x \rangle a)$. In particular, given such a y we can abstract again on y and form $\langle y \rangle (\langle x \rangle a \mathbin{@} y) = \langle x \rangle a$.

1.2 Categorical preliminaries

In all what follows and if not specified, the category $\mathbf{C}$ is either **Set** or **Nom**.

Given an endofunctor $F : \mathbf{C} \to \mathbf{C}$, an *$F$-algebra* (A, α) is an object $A \in \mathbf{C}$ together with an arrow $\alpha : FA \to A$. An algebra morphism $(A, \alpha) \to (B, \beta)$ is an arrow $f : A \to B$ such that $\beta \circ Ff = f \circ \alpha$ in $\mathbf{C}$. This defines a category of F-algebras. When this category has an initial object, it is called the *initial algebra* of F and is denoted by $(\mu X.FX, \mathsf{fold}_F)$, or only $\mu X.FX$ when there is no ambiguity. Dualising all these definitions, one obtains a notion of *terminal coalgebra* for an endofunctor F, denoted by $\nu X.FX$ when it exists.

Lambek's lemma [20] states that the arrows supporting initial algebras and terminal coalgebras are isomorphisms. This implies that an initial algebra is a coalgebra, and that a terminal coalgebra is an algebra. As a consequence, there is a canonical morphism $\mu X.FX \rightarrowtail \nu X.FX$.

[2] So far, we do not precise the cardinality of $\mathcal{V}$. In all what follows, $\mathcal{V}$ can be countable or uncountable, if not specified.

All the functors that we will consider will have a polynomial shape that makes them *ω-cocontinuous*, *i.e.* they commute to colimits of ω-chains[3]. This entails the existence of their initial algebra. Given an ω-cocontinuous bifunctor $F : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, one can take the initial algebra in the first variable: this gives rise to an ω-cocontinuous functor $\mathbb{\mu}X.F(X,-) : \mathbf{C} \to \mathbf{C}$.

Lemma 1 (diagonal identity) *Given an ω-cocontinuous functor $F : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$,*

$$\mathbb{\mu}Y.\mathbb{\mu}X.F(X,Y) = \mathbb{\mu}Z.F(Z,Z)$$

in the category of $F\Delta$-algebras, where $\Delta : X \mapsto (X,X)$ is the diagonal functor.

This lemma is standard, and has been first proved in a categorical setting by Lehmann and Smyth [21]. An alternative proof is proposed in the appendices of the long version of this abstract.

1.3 MBS and raw terms

Binding signatures [27, 13] provide a general description of term (co)algebras with binding operators. Let us quickly recall their main properties. A *binding signature* (BS) is a couple (Σ, ar) where Σ is a set at most countable of constructors, and $\mathsf{ar} : \Sigma \to \mathbb{N}^*$ is a function indicating the binding arity of each argument of each constructor. Given a BS (Σ, ar), its term functor $\mathcal{F}_\Sigma : \mathbf{C} \to \mathbf{C}$ is defined by

$$\mathcal{F}_\Sigma X := \mathcal{V} + \coprod_{\substack{\mathsf{cons} \in \Sigma \\ \mathsf{ar}(\mathsf{cons}) = (n_1, \dots, n_k)}} \prod_{i=1}^{k} \mathcal{V}^{n_i} \times X.$$

The sets of raw (*i.e.* not quotiented by α-equivalence) finite and infinitary terms on (Σ, ar) are then defined by $\mathcal{T}_\Sigma := \mathbb{\mu}X.\mathcal{F}_\Sigma X$ and $\mathcal{T}_\Sigma^\infty := \mathbb{\nu}X.\mathcal{F}_\Sigma X$ (in **Set**). Notice that these (co)algebras always exist, thanks to the polynomial shape of $\mathcal{F}_\Sigma$. A typical example is the signature: $\Sigma_\lambda := \{\lambda, @\}$ with $\mathsf{ar}(\lambda) := (1)$ and $\mathsf{ar}(@) := (0,0)$, such that $\mathcal{T}_\lambda$ is the algebra Λ of all finite λ-terms, and $\mathcal{T}_\lambda^\infty$ is the coalgebra Λ^{111} of all (full) infinitary λ-terms.

We want to tweak this definition in order to be able to design mixed inductive-coinductive data types with binding[4]. An elementary example of such a mixing (with no binding) is the type of *right-infinitary* binary trees: the set of all infinitary binary trees such that each infinite branch contains infinitely many right edges. This type can be defined as $\mathbb{\nu}Y.\mathbb{\mu}X.1 + X \times Y$ in **Set**. Our aim is to be able to express such a construction when some constructors bind variables (and then investigate the quotient by α-equivalence).

Definition 2 (mixed binding signature) *A* mixed binding signature *(MBS) is a couple (Σ, ar) where Σ is a set at most countable of constructors, and $\mathsf{ar} : \Sigma \to (\mathbb{N} \times \mathbb{B})^*$ is an arity function.*

$\mathbb{B}$ denotes the set of booleans: each argument of each constructor is marked with a boolean describing its (co)inductive behaviour. This intuition is driving the following definitions, that allow to define *mixed* terms on a MBS.

[3] What we have in mind is the naive notion of polynomial, as considered for instance by Adámek, Milius, and Moss [1] or Métayer [23]. In particular, the broader notion known as *polynomial functors* encompasses functors with infinite powers, which prevents ω-cocontinuity in general. See Kock [17, § 1.7.3] for a discussion.

[4] Existing generalisations of binding signatures go way beyond our modest extension, that could certainly be reformulated in a broader setting — see *e.g.* Power [28] (whose work subsumes both Fiore, Plotkin, and Turi's binding algebras and Gabbay and Pitts' nominal sets), as well as Adámek, Milius, and Velebil [2] or Arkor [4]. However, it is not completely clear to us whether these abstract frameworks encompass coinductive syntax in the way we want to construct it, without any further work.

$$\frac{x \in \mathcal{V}}{x \in \mathcal{T}_\Sigma^\infty} \qquad \frac{t \in \mathcal{T}_\Sigma^\infty}{\blacktriangleright_0\, t \in \mathcal{T}_\Sigma^\infty} \qquad \frac{\!\!\frac{t \in \mathcal{T}_\Sigma^\infty}{\blacktriangleright_1\, t \in \mathcal{T}_\Sigma^\infty}\!\!}{}$$

$$\frac{\bar{x_1} \in \mathcal{V}^{n_1} \quad \cdots \quad \bar{x_k} \in \mathcal{V}^{n_k} \quad \blacktriangleright_{b_1} t_1 \in \mathcal{T}_\Sigma^\infty \quad \cdots \quad \blacktriangleright_{b_k} t_k \in \mathcal{T}_\Sigma^\infty}{\mathsf{cons}\,(\bar{x_1}.t_1, \ldots, \bar{x_k}.t_k) \in \mathcal{T}_\Sigma^\infty}$$

for each $\mathsf{cons} \in \Sigma$, having $\mathsf{ar}(\mathsf{cons}) = ((n_1, b_1), \ldots, (n_k, b_k))$

Figure 1: Formal system defining $\mathcal{T}_\Sigma^\infty$ for a MBS (Σ, ar). The simple rules are inductive, the double one is coinductive; for similar systems, see [12, 11].

$$\frac{x \in \mathcal{V}}{x \in \Lambda^{001}} \qquad \frac{\!\!\frac{M \in \Lambda^{001}}{\blacktriangleright\, M \in \Lambda^{001}}\!\!}{} \qquad \frac{x \in \mathcal{V} \quad M \in \Lambda^{001}}{\lambda(x.M) \in \Lambda^{001}} \qquad \frac{M \in \Lambda^{001} \quad \blacktriangleright\, N \in \Lambda^{001}}{@(M, N) \in \Lambda^{001}}$$

Figure 2: A simplified mixed formal system defining Λ^{001}.

Definition 3 (term functor of a MBS) *The polynomial* term functor *associated to* (Σ, ar) *is the* $\mathbf{C}$*-bifunctor* $\mathcal{F}_\Sigma$ *defined by:*

$$\mathcal{F}_\Sigma(X, Y) := \mathcal{V} + \coprod_{\substack{\mathsf{cons} \in \Sigma \\ \mathsf{ar}(\mathsf{cons}) = ((n_1, b_1), \ldots, (n_k, b_k))}} \prod_{i=1}^{k} \mathcal{V}^{n_i} \times \pi_{b_i}(X, Y)$$

where π_0 *and* π_1 *are the projections.*

Lemma 1 ensures that there is a unique notion of "fully initial" algebra on a bifunctor, hence the definition of raw terms on a MBS.

Definition 4 (raw terms on a MBS) *The sets* $\mathcal{T}_\Sigma$ *of* raw finite terms *and* $\mathcal{T}_\Sigma^\infty$ *of* raw mixed terms *on* (Σ, ar) *are defined by:*

$$\mathcal{T}_\Sigma := \mathbb{\mu} Z.\mathcal{F}_\Sigma(Z, Z) \qquad \mathcal{T}_\Sigma^\infty := \mathbb{\nu} Y.\mathbb{\mu} X.\mathcal{F}_\Sigma(X, Y).$$

Notation 5 *We can describe* $\mathcal{T}_\Sigma^\infty$ *by means of a (mixed) formal system of derivation rules, as proposed in fig. 1. We use the symbols* $\blacktriangleright_0$ *and* $\blacktriangleright_1$ *to distinguish between the inductive and coinductive calls.* $\blacktriangleright_1$ *is usually called the* later *modality [24, 3]; a derivation of* $\blacktriangleright_1 P$ *is a derivation of* P *under an additional coinductive guard. The modality* $\blacktriangleright_0$ *could be omitted, but we write it to keep the notations symmetric.*

Example 6 (mixed infinitary λ-terms) *For* $a, b, c \in \mathbb{B}$*, the* MBS $(\Sigma_\lambda, \mathsf{ar}_{abc})$ *is defined by:*

$$\Sigma_\lambda := \{\lambda, @\} \qquad \mathsf{ar}_{abc}(\lambda) := ((1, a)) \qquad \mathsf{ar}_{abc}(@) := ((0, b), (0, c)).$$

For any a, b, c*,* $\mathcal{T}_{\lambda abc}$ *is the algebra* Λ *of finite* λ*-terms and* $\mathcal{T}_{\lambda abc}^\infty$ *is the coalgebra of* abc*-infinitary* λ*-terms. For instance, the 001-infinitary* λ*-terms are described by fig. 2.*

1.4 Metric completion

Take $\mathbf{C}$ to be **Set**. Following a standard path, we define the Arnold-Nivat metric [5] on both $\mathcal{T}_\Sigma$ and $\mathcal{T}_\Sigma^\infty$. To do so, we use the following notion of truncation, adapted to a mixed inductive-coinductive setting.

Definition 7 (truncation) *Given an integer $n \in \mathbb{N}$ and a term t in either $\mathcal{T}_\Sigma$ or $\mathcal{T}_\Sigma^\infty$, the* mixed truncation *at depth n of t is the object*[5] $\lfloor t \rfloor_n \in (\mu X.\mathcal{F}_\Sigma(X,-))^n 1$ *defined by induction by:*

$$\begin{aligned} \lfloor t \rfloor_0 &:= * \\ \lfloor x \rfloor_{n+1} &:= x \\ \lfloor \mathsf{cons}\,(\bar{x_1}.t_1,\dots,\bar{x_k}.t_k) \rfloor_{n+1} &:= \mathsf{cons}\,(\bar{x_1}.\lfloor t_1 \rfloor_{n+1-b_1},\dots,\bar{x_k}.\lfloor t_k \rfloor_{n+1-b_k}) \end{aligned}$$

where $b_i = \pi_1 \mathsf{ar}(\mathsf{cons})_i$.

Notice that the definition is by double induction, on n and on t (even if the latter is taken in $\mathcal{T}_\Sigma^\infty$): in the inductive arguments of cons we proceed by induction on t, in its coinductive arguments we proceed by induction on n.

Definition 8 (Arnold-Nivat metric) *The* Arnold-Nivat metric *on $\mathcal{T}_\Sigma$ and $\mathcal{T}_\Sigma^\infty$ is the mapping $\mathbb{d} : \mathcal{T}_\Sigma^\infty \times \mathcal{T}_\Sigma^\infty \to \mathbb{R}_+$ defined by $\mathbb{d}(t,u) := \inf\{\, 2^{-n} \,|\, n \in \mathbb{N},\ \lfloor t \rfloor_n = \lfloor u \rfloor_n \,\}$.*

The unique notation is unambiguous, since the canonical inclusion $\mathcal{T}_\Sigma \rightarrowtail \mathcal{T}_\Sigma^\infty$ preserves the truncations. The following fact is a translation of [7, Th. 3.2], using lemma 1. It expresses the equivalence of our coinductive definition of $\mathcal{T}_\Sigma^\infty$ and the historical topological point of view [16].

Lemma 9 *$\mathcal{T}_\Sigma^\infty$ is the Cauchy completion of $\mathcal{T}_\Sigma$ with respect to $\mathbb{d}$. Furthermore, the completion is carried by the canonical arrow $\mathcal{T}_\Sigma \rightarrowtail \mathcal{T}_\Sigma^\infty$.*

Example 10 *The eight Arnold-Nivat metrics $\mathbb{d}^{abc}$ corresponding to the signatures from example 6 are exactly those considered in the original definition of infinitary λ-calculi [16]. Hence our coinductive definition of Λ^{abc} coincides with the historical, topological definition.*

1.5 α-equivalence

α-equivalence is the equivalence relation generated on some term (co)algebra by renaming all bound variables. Let us recall how this can be reformulated in a nominal setting (for finite terms): given a BS or a MBS (Σ, ar), the finite term algebra $\mathcal{T}_\Sigma$ can be endowed with a $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-action $\cdot$ inductively defined by:

$$\begin{aligned} \sigma \cdot x &:= \sigma(x) \\ \sigma \cdot \mathsf{cons}\,(\bar{x_1}.t_1,\dots) &:= \mathsf{cons}\,(\sigma(\bar{x_1}).\sigma \cdot t_1,\dots), \end{aligned} \tag{1}$$

where permutations act pointwise on the sequences $\bar{x_i}$. This defines a nominal set $(\mathcal{T}_\Sigma, \cdot)$. The α-equivalence relation is then defined by:

$$\frac{}{x =_\alpha x} \qquad \frac{((\bar{x_i}\ \bar{z_i}) \cdot t_i =_\alpha (\bar{y_i}\ \bar{z_i}) \cdot u_i \text{ for fresh } \bar{z_i})_{i=1}^k}{\mathsf{cons}\,(\bar{x_1}.t_1,\dots) =_\alpha \mathsf{cons}\,(\bar{y_1}.u_1,\dots)}$$

where the permutation $(\bar{x_i}\ \bar{z_i})$ is the composition of the transpositions $(x_i\ z_i)$. This equivalence relation is compatible with $\cdot$, thus there is an induced nominal structure $(\mathcal{T}_\Sigma/{=_\alpha}, \cdot)$.

An important theorem by Gabbay and Pitts [14, 26, Th. 8.15] can be straightforwardly transported to our mixed setting.

[5] We try not to be too formal here. In the following we manipulate truncations as if they were finite terms on $\Sigma \cup \{*\}$, where $*$ is a new constant.

Definition 11 (quotient term functor of a MBS) *The polynomial* quotient term functor *associated to* (Σ, ar) *is the* **Nom***-bifunctor* Q_Σ *defined by:*

$$Q_\Sigma(X,Y) := \mathcal{V} + \coprod_{\substack{\mathsf{cons}\in\Sigma \\ \mathsf{ar}(\mathsf{cons})=((n_1,b_1),\dots,(n_k,b_k))}} \prod_{i=1}^{k} [\mathcal{V}]^{n_i}\pi_{b_i}(X,Y).$$

Theorem 12 (nominal algebraic types on a MBS) *Given a* MBS (Σ, ar)*, then* $\mathcal{T}_\Sigma = \mathbb{\mu}Z.\mathcal{F}_\Sigma(Z,Z)$ *and* $\mathcal{T}_\Sigma/{=_\alpha} = \mathbb{\mu}Z.Q_\Sigma(Z,Z)$ *in* **Nom**.

The first identity might seem tautologic because of the overloaded the notation $\mathcal{F}_\Sigma$; if we distinguish between $\mathcal{F}_\Sigma^{\mathbf{Set}}$ and $\mathcal{F}_\Sigma^{\mathbf{Nom}}$ it becomes $(\mathbb{\mu}Z.\mathcal{F}_\Sigma^{\mathbf{Set}}(Z,Z),\cdot) = \mathbb{\mu}Z.\mathcal{F}_\Sigma^{\mathbf{Nom}}(Z,Z)$, where the former nominal structure was built in eq. (1).

1.6 Towards commutation (or not)

For now, we have built the following diagram (in **Set**):

$$\begin{array}{ccc} \overset{U(\mathbb{\mu}Z.\mathcal{F}_\Sigma(Z,Z))}{\underset{\mathbb{\mu}Z.\mathcal{F}_\Sigma(Z,Z)}{\mathcal{T}_\Sigma}} & \xrightarrow{\text{compl.}} & \overset{\mathbb{\nu}Y.\mathbb{\mu}X.\mathcal{F}_\Sigma(X,Y)}{\mathcal{T}_\Sigma^\infty} \\ \downarrow & & \\ \underset{U(\mathbb{\mu}Z.Q_\Sigma(Z,Z))}{\mathcal{T}_\Sigma/{=_\alpha}} & & \end{array} \tag{2}$$

The sets are annotated with their descriptions as (co)algebras in **Set** and in **Nom** (U is the forgetful functor **Nom** $\to$ **Set**). The horizontal arrow is the metric completion given by lemma 9, the vertical surjection is the quotient by α-equivalence given by theorem 12. Our goal is to close the square with an object containing α-equivalence classes of mixed terms; we hope to obtain a nominal presentation of this object. To do so, we keep adapting the definitions of [19] to our mixed setting:

- $\mathcal{T}_\Sigma^\infty$ can be equipped with a $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-action in the same way as we did in eq. (1) for the finitary setting, by just making the definition coinductive; however, this does not define a nominal set any more since some infinitary terms are not finitely supported (the support of a term being the set of the variables occurring in it).
- As a consequence, we cannot directly use a nominal set structure to extend the definition of α-equivalence to $\mathcal{T}_\Sigma^\infty$. Instead, we lift the α-equivalence of $\mathcal{T}_\Sigma$ by using the truncations: two mixed terms $t,u \in \mathcal{T}_\Sigma^\infty$ are then said to be α-equivalent if $\forall n \in \mathbb{N},\ \lfloor t \rfloor_n =_\alpha \lfloor u \rfloor_n$.
- We also define a metric on $\mathcal{T}_\Sigma/{=_\alpha}$ as in definition 8: $\mathbb{d}_\alpha(t,u) := \inf\{2^{-n} \mid n \in \mathbb{N},\ \lfloor t \rfloor_n =_\alpha \lfloor u \rfloor_n\}$. Then $(\mathcal{T}_\Sigma/{=_\alpha})^\infty$ is the metric completion of $\mathcal{T}_\Sigma/{=_\alpha}$ with respect to $\mathbb{d}_\alpha$.

These constructions extend diagram 2 as follows:

$$\begin{array}{ccc} \overset{U(\mathbb{\mu}Z.\mathcal{F}_\Sigma(Z,Z))}{\underset{\mathbb{\mu}Z.\mathcal{F}_\Sigma(Z,Z)}{\mathcal{T}_\Sigma}} & \xrightarrow{\text{compl.}} & \overset{\mathbb{\nu}Y.\mathbb{\mu}X.\mathcal{F}_\Sigma(X,Y)}{\mathcal{T}_\Sigma^\infty} \\ \downarrow & & \downarrow \\ & & \mathcal{T}_\Sigma^\infty/{=_\alpha} \\ & & \downarrow ? \\ \underset{U(\mathbb{\mu}Z.Q_\Sigma(Z,Z))}{\mathcal{T}_\Sigma/{=_\alpha}} & \xrightarrow{\text{compl.}} & (\mathcal{T}_\Sigma/{=_\alpha})^\infty \end{array} \tag{3}$$

The existence of an inclusion $\overset{?}{\hookrightarrow}$ is straightforward, but we would like an isomorphism instead. Unfortunately, it is not the case in general, unless the signature is trivial in the following meaning.

Definition 13 (non-trivial MBS) *A MBS* (Σ, ar) *is* non-trivial *if there are constructors* $\mathsf{lam}, \mathsf{node}, \mathsf{dig} \in \Sigma$ *such that:*

1. lam *has a binding argument,* i.e. $\pi_0(\mathsf{ar}(\mathsf{lam})_i) \geq 1$ *for some index* i*;*
2. node *has at least two arguments,* i.e. $\mathsf{ar}(\mathsf{node})$ *is of length greater than 2;*
3. dig *has a coinductive argument,* i.e. $\pi_1(\mathsf{ar}(\mathsf{dig})_i) = 1$ *for some index* i*.*

If the signature is trivial, it does not make sense to consider all the machinery defined here: if there is no binder then $=_\alpha$ amounts to equality, if there are only unary and constant constructors then there is at most one variable in each term, and if there is no coinductive constructor then the metric is discrete. In all three cases, $(\mathcal{T}_\Sigma^\infty/{=_\alpha}) \cong (\mathcal{T}_\Sigma/{=_\alpha})^\infty$ for degenerate reasons. Otherwise, the cardinality of $\mathcal{V}$ is determining, as theorem 14 shows.

Theorem 14 *Let* (Σ, ar) *be a non-trivial MBS. Then* $(\mathcal{T}_\Sigma^\infty/{=_\alpha}) \cong (\mathcal{T}_\Sigma/{=_\alpha})^\infty$ *iff* $\mathcal{V}$ *is uncountable.*

Our goal is not really fulfilled: we have a commutative square only if $\mathcal{V}$ is uncountable, which is not satisfactory in practice since implementation concerns suggest to consider countably many variables. In addition, none of the sets involved can be endowed with a reasonable nominal structure.

2 The nominal coalgebra of α-equivalence classes of mixed terms

In this second part, we show that Kurz, Petrişan, Severi, and de Vries' theorem has a mixed counterpart. Then we use this result to define substitution on mixed terms by nested recursion and corecursion.

2.1 Nominal mixed types

The following structure is, once again, extended to the setting of mixed terms:

- Given a set S equipped with a $\mathfrak{S}_{\mathsf{fs}}(\mathcal{V})$-action, S_{fs} is the subset of finitely supported elements of S. It carries a nominal set structure. In particular $(\mathcal{T}_\Sigma^\infty)_{\mathsf{fs}}$ is the nominal set of the finitely supported raw terms in $\mathcal{T}_\Sigma^\infty$, and $(\mathcal{T}_\Sigma/{=_\alpha})_{\mathsf{fs}}^\infty$ is the nominal set of finitely supported α-equivalence classes in $(\mathcal{T}_\Sigma/{=_\alpha})^\infty$.
- $(\mathcal{T}_\Sigma^\infty)_{\mathrm{ffv}}$ denotes the set of infinitary terms having finitely many free variables.

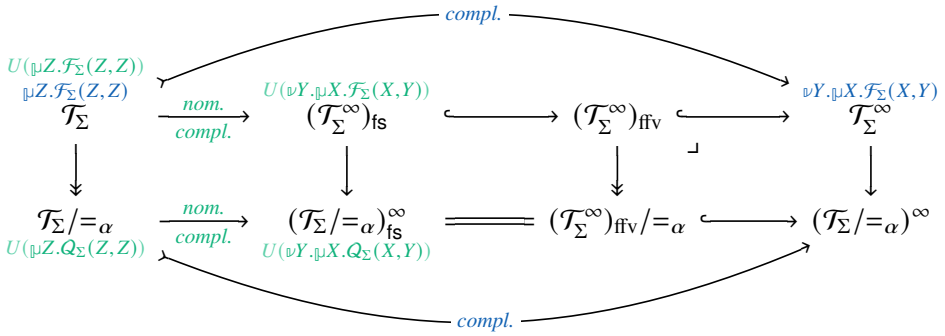
Recall also that given a nominal metric space (*i.e.* a nominal space equipped with an equivariant metric), its nominal metric completion is built by adding the limits of all finitely supported Cauchy sequences (*i.e.* sequences of terms such that their supports are all contained in a common finite set).

Let us state the main theorem of our fanfiction without delay, as well as its crucial corollary.

Theorem 15 (nominal mixed terms on a MBS) *Let MBS* (Σ, ar) *be a MBS. Then:*

1. *The nominal set* $(\mathcal{T}_\Sigma^\infty)_{\mathsf{fs}}$ *is the nominal metric completion of* $\mathcal{T}_\Sigma$*, as well as the terminal coalgebra* $\nu Y.\mathbb{\mu} X.\mathcal{F}_\Sigma(X,Y)$*.*
2. *Similarly, the nominal set* $(\mathcal{T}_\Sigma/{=_\alpha})_{\mathsf{fs}}^\infty$ *is the nominal metric completion of* $\mathcal{T}_\Sigma/{=_\alpha}$*, as well as the terminal coalgebra* $\nu Y.\mathbb{\mu} X.\mathcal{Q}_\Sigma(X,Y)$*.*

3. *The following diagram commutes in* **Set**:



Corollary 16 *The nominal set* $(\mathcal{T}_\Sigma^\infty)_{\mathrm{ffv}}/{=_\alpha}$ *is the terminal coalgebra* $\mathbb{\nu}Y.\mathbb{\mu}X.\mathcal{Q}_\Sigma(X,Y)$.

These results are direct counterparts to Remark 5.30, Theorem 5.34 and Corollary 5.35 from [19], and the diagram we provide is exactly the same as their diagram 5.20. The only difference here is that we take the terminal coalgebra of $\mathbb{\mu}X.\mathcal{F}_\Sigma(X,-)$ and $\mathbb{\mu}X.\mathcal{Q}_\Sigma(X,-)$, instead of $\mathcal{F}_\Sigma$ and $\mathcal{Q}_\Sigma$ themselves. What we need to show is that all the technical developments of [19] remain applicable[6].

Lemma 17 *Let* $F : \mathbf{Nom}\times\mathbf{Nom} \to \mathbf{Nom}$ *be polynomial in the following sense: there are a countable set* I *and families* $\{k_i \in \mathbb{N} \mid i \in I\}$, $\left\{m_{ij} \in \mathbb{N} \middle|\, {}^{i\in I}_{1\le j\le k_i}\right\}$ *and* $\left\{b_{ij} \in \mathbb{B} \middle|\, {}^{i\in I}_{1\le j\le k_i}\right\}$ *such that*

$$F = K + \coprod_{i\in I}\prod_{j=1}^{k_i} M^{m_{ij}}\pi_{b_{ij}}$$

where π_0 *and* π_1 *denote the projections,* $M : \mathbf{Nom} \to \mathbf{Nom}$ *is a fixed functor commuting to directed colimits, and* K *is a fixed constant functor. Then* $\mathbb{\mu}X.F(X,-)$ *exists and can be obtained from the following grammar (up to isomorphism):*

$$G \quad ::= \quad \mathrm{id} \mid K \mid MG \mid \coprod G \mid G\times G \qquad (\Gamma_1)$$

where $\coprod$ *denotes at most countable coproducts.*

Using the lemma, the proof of theorem 15 and corollary 16 is straightforward: taking K to be the constant functor $\mathcal{V}$, and M to be either $\mathcal{V}\times -$ or $[\mathcal{V}]$, we just showed that $\mathbb{\mu}X.\mathcal{F}_\Sigma(X,-)$ and $\mathbb{\mu}X.\mathcal{Q}_\Sigma(X,-)$ fulfill the requirements of [19, Prop. 5.6].

Example 18 *The nominal set* $\Lambda^{001}_{\mathrm{ffv}}/{=_\alpha}$ *of* α*-equivalence classes of 001-infinitary* λ*-terms having finitely many free variables is the terminal coalgebra* $\mathbb{\nu}Y.\mathbb{\mu}X.\mathcal{V} + [\mathcal{V}]X + X\times Y$.

2.2 Capture-avoiding substitution for mixed types

We fix a MBS (Σ, ar), and we write $\mathcal{T}^\infty_\alpha$ for $\mathbb{\nu}Y.\mathbb{\mu}X.\mathcal{Q}_\Sigma(X,Y)$. We want to define capture-avoiding substitution as a map $\mathsf{subst} : \mathcal{T}^\infty_\alpha \times \mathcal{V} \times \mathcal{T}^\infty_\alpha \to \mathcal{T}^\infty_\alpha$ in **Nom**.

As in [19, Def. 6.2], we shall use the corecursion principle of [22, Lem. 2.1]. However, this is not enough any more: we also have to scan the inductive structure separating two coinductive constructors and, since this structure may contain variables (in fact *all* the variables appear in these "inductive layers"), perform substitution recursively on it too.

[6]During the writing of this paper, we came up with an explicit construction of our mixed terms as purely coinductive terms on a modified binding signature. From this, one gets an alternative proof of the theorem. Even if it is not useful for our purposes, we provide this construction in the appendices of the long version of this abstract, just in case.

Notation 19 *When we consider a coproduct $A+B$, we write* inl *and* inr *for the left and right injections. Similarly, we write* invar *and* incons *the injections in initial algebras of the form $\mathbb{\mu}X.Q_\Sigma(X,Y)$. We omit the composition by* fold *for the sake of readability.*

Notation 20 *It is easy to show that* **Nom***-endofunctors obtained from (grammar Γ_1) are strong, hence we denote:*

- *by $\tau_{A,B} : [\mathcal{V}]A\times B \to [\mathcal{V}](A\times B)$ the strength defined by $(\langle x\rangle a, b) \mapsto \langle z\rangle(\langle x\rangle a \mathbin{@} z, b)$,*
- *by τ the strength $\tau_{\mathcal{T}_\alpha^\infty, \mathcal{V}\times\mathcal{T}_\alpha^\infty}$ and by $\tau_n : [\mathcal{V}]^n\mathcal{T}_\alpha^\infty\times\mathcal{V}\times\mathcal{T}_\alpha^\infty \to [\mathcal{V}]^n(\mathcal{T}_\alpha^\infty\times\mathcal{V}\times\mathcal{T}_\alpha^\infty)$ its iteration,*
- *by $\bar\tau$ the strength generated for $\mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty + -)$.*

Using these notations, we are finally able to define capture-avoiding substitution.

Definition 21 (capture-avoiding substitution) Capture-avoiding substitution *is the map* subst *defined by:*

$$\begin{array}{ccc}
\mathcal{T}_\alpha^\infty\times\mathcal{V}\times\mathcal{T}_\alpha^\infty & \xrightarrow{\quad\text{subst}\quad} & \mathcal{T}_\alpha^\infty \\
{\scriptstyle \text{unfold}\times\mathcal{V}\times\mathcal{T}_\alpha^\infty}\downarrow & & \\
\mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty)\times\mathcal{V}\times\mathcal{T}_\alpha^\infty & & \downarrow{\scriptstyle \text{unfold}} \\
{\scriptstyle h'}\downarrow & & \\
\mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty + \mathcal{T}_\alpha^\infty\times\mathcal{V}\times\mathcal{T}_\alpha^\infty) & \xrightarrow{\mathbb{\mu}X.Q_\Sigma(X,\text{id+subst})} & \mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty)
\end{array}$$

where h' is recursively defined by:

$$\begin{aligned}
&(\mathsf{invar}(x),x,u) \mapsto \mathbb{\mu}X.Q_\Sigma(X,\mathsf{inl})(\mathsf{unfold}(u)) \\
&(\mathsf{invar}(y),x,u) \mapsto \mathsf{invar}(y) \qquad\qquad \textit{for } y\neq x \\
&\left(\mathsf{incons}\begin{pmatrix}\vdots\\ \langle y_{i,1}\rangle\dots\langle y_{i,n_i}\rangle t_i,\\ \vdots\\ \langle y_{j,1}\rangle\dots\langle y_{j,n_j}\rangle t_j\\ \vdots\end{pmatrix},x,u\right) \mapsto \mathbb{\mu}X.Q_\Sigma(X,\mathsf{inr})\begin{pmatrix}\vdots\\ \langle y_{i,1}\rangle\dots\langle y_{i,n_i}\rangle h'(t_i,x,u),\\ \vdots\\ \tau_{n_j}(\langle y_{j,1}\rangle\dots\langle y_{j,n_j}\rangle t_j,x,u)\\ \vdots\end{pmatrix}
\end{aligned}$$

where i (resp. j) represents any index such that $b_i = 0$ (resp. $b_j = 1$), i.e. *any inductive (resp. coinductive) position of* cons*), and where the representatives are taken so that $\forall k\in[0,n_i]$, $y_{i,k}\#x$ and $y_{i,k}\#u$.*

In fact the validity of the recursive definition of h' is not immediate; in particular, it is not straightforwardly implied by Pitts' recursion theorem for nominal algebras [25, Thm. 5.1] (see also [26, § 8.5] for lighter presentation). This is due to the fact that h' is not purely inductive, it also inserts τ_{n_j}'s in coinductive positions (which amounts to modifying the constructors of the local induction step). This is why a rigorous definition of h' relies on the following decomposition into a purely inductive h, followed by

some work on the coinductive structure of the terms:

$$
\begin{array}{rc}
 & \mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty)\times\mathcal{V}\times\mathcal{T}_\alpha^\infty \\
 & \Big\downarrow{\scriptstyle \mathbb{\mu}X.Q_\Sigma(X,\mathsf{inr})\times\mathcal{V}\times\mathcal{T}_\alpha^\infty} \\
 & \mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty+\mathcal{T}_\alpha^\infty)\times\mathcal{V}\times\mathcal{T}_\alpha^\infty \\
h' := & \Big\downarrow{\scriptstyle h\times\mathcal{V}\times\mathcal{T}_\alpha^\infty} \\
 & \mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty+\mathcal{T}_\alpha^\infty)\times\mathcal{V}\times\mathcal{T}_\alpha^\infty \\
 & \Big\downarrow{\scriptstyle \bar{\tau}} \\
 & \mathbb{\mu}X.Q_\Sigma(X,\mathcal{T}_\alpha^\infty+\mathcal{T}_\alpha^\infty\times\mathcal{V}\times\mathcal{T}_\alpha^\infty)
\end{array}
$$

where $h : (t,x,u) \mapsto h_{x,u}(t)$ is uniquely defined by recursion by

$$
\begin{array}{ll}
\mathsf{invar}(x) \mapsto \mathbb{\mu}X.Q_\Sigma(X,\mathsf{inl})(\mathsf{unfold}(u)) & \\
\mathsf{invar}(y) \mapsto \mathsf{invar}(y) & \text{for } y \neq x \\
\mathsf{incons}\begin{pmatrix} \vdots \\ \langle y_{i,1}\rangle\ldots\langle y_{i,n_i}\rangle t_i, \\ \vdots \\ \langle y_{j,1}\rangle\ldots\langle y_{j,n_j}\rangle T_j \\ \vdots \end{pmatrix} \mapsto \mathsf{incons}\begin{pmatrix} \vdots \\ \langle y_{i,1}\rangle\ldots\langle y_{i,n_i}\rangle h_{x,u}(t_i), \\ \vdots \\ \langle y_{j,1}\rangle\ldots\langle y_{j,n_j}\rangle T_j \\ \vdots \end{pmatrix} &
\end{array}
$$

under the hypotheses and notations of definition 21, that ensure that the "freshness condition for binders" of Pitt's recursion theorem is satisfied, hence the well-definedness of h.

Example 22 *Let us describe what h' looks like when $\mathcal{T}_\alpha^\infty$ is $\Lambda_{\mathrm{ffv}}^{001}/{=_\alpha}$:*

$$
\begin{array}{rll}
(x,x,N) \mapsto \mathbb{\mu}X.Q_{\lambda 001}(X,\mathsf{inl})(\mathsf{unfold}(N)) & & \\
(y,x,N) \mapsto y & & \textit{for } y \neq x \\
(\lambda(y.M),x,N) \mapsto \mathbb{\mu}X.Q_{\lambda 001}(X,\mathsf{inr})(\lambda(y.h(M,x,N))) & & \textit{for } y \neq x \textit{ and } y \notin \mathrm{fv}(N) \\
(@(M_0,M_1),x,N) \mapsto \mathbb{\mu}X.Q_{\lambda 001}(X,\mathsf{inr})\,(@\,(h(M_0,x,N),(M_1,x,N)))\,, & &
\end{array}
$$

where we omitted the injections. Finally we obtain the expected recursive-corecursive definition of capture-avoiding substitution:

$$
\begin{array}{rll}
\mathsf{subst}(x,x,N) := N & & \\
\mathsf{subst}(y,x,N) := y & & \textit{for } y \neq x \\
\mathsf{subst}(\lambda(y.M),x,N) := \lambda(y.\mathsf{subst}(M,x,N)) & & \textit{for } y \neq x \textit{ and } y \notin \mathrm{fv}(N) \\
\mathsf{subst}(@(M_0,M_1),x,N) := @(\mathsf{subst}(M_0,x,N),\mathsf{subst}(M_1,x,N)). & &
\end{array}
$$

References

[1] Jiří Adámek, Stefan Milius & Lawrence S. Moss (2018): *Fixed points of functors*. *Journal of Logical and Algebraic Methods in Programming* 95, pp. 41–81, doi:10.1016/j.jlamp.2017.11.003.

[2] Jiří Adámek, Stefan Milius & Jiří Velebil (2009): *Semantics of Higher-Order Recursion Schemes*. In: *CALCO* 2009: *Algebra and Coalgebra in Computer Science*, pp. 49–63, doi:10.1007/978-3-642-03741-2_5.

[3] Andrew W. Appel, Paul-André Melliès, Christopher D. Richards & Jérôme Vouillon (2007): *A very modal model of a modern, major, general type system*. *ACM SIGPLAN Notices* 42(1), pp. 109–122, doi:10.1145/1190215.1190235.

[4] Nathanael Arkor (2022): *Monadic and Higher-Order Structure*. phdthesis, University of Cambridge, doi:10.17863/CAM.86347.

[5] André Arnold & Maurice Nivat (1980): *The metric space of infinite trees. Algebraic and topological properties*. *Fundamenta Informaticae* 3(4), pp. 445–475, doi:10.3233/fi-1980-3405.

[6] Henk P. Barendregt (1984): *The Lambda Calculus. Its Syntax and Semantics*, 2 edition. *Studies in Logic and the Foundations of Mathematics* 103, Elsevier Science, doi:10.1016/B978-0-444-87508-2.50006-X.

[7] Michael Barr (1993): *Terminal coalgebras in well-founded set theory*. *Theoretical Computer Science* 114(2), pp. 299–315, doi:10.1016/0304-3975(93)90076-6.

[8] Alessandro Berarducci (1996): *Infinite λ-calculus and non-sensible models*. In: *Logic and Algebra*, Routledge, pp. 339–377, doi:10.1201/9780203748671-17.

[9] Nicolaas Govert de Bruijn (1972): *Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with applications to the Church-Rosser Theorem*. *Indagationes Mathematicæ* 34, doi:10.1016/1385-7258(72)90034-0.

[10] Rémy Cerda (2024): *Taylor Approximation for Infinitary λ-Calculi*. Ph.D. thesis, Aix-Marseille University.

[11] Rémy Cerda & Lionel Vaux Auclair (2023): *Finitary Simulation of Infinitary β-Reduction via Taylor Expansion, and Applications*. arXiv:2211.05608.

[12] Ugo Dal Lago (2016): *Infinitary λ-Calculi from a Linear Perspective*. arXiv:1604.08248. Long version of a *LICS* paper.

[13] Marcelo Fiore, Gordon Plotkin & Daniele Turi (1999): *Abstract syntax and variable binding*. In: 14*th Symposium on Logic in Computer Science*, doi:10.1109/lics.1999.782615.

[14] Murdoch J. Gabbay & Andrew M. Pitts (2002): *A New Approach to Abstract Syntax with Variable Binding*. *Formal Aspects of Computing* (13), pp. 341–363, doi:10.1007/s001650200016.

[15] André Hirschowitz, Tom Hirschowitz, Ambroise Lafont & Marco Maggesi (2022): *Variable binding and substitution for (nameless) dummies*. In: *FoSSaCS* 2022: *Foundations of Software Science and Computation Structures*, pp. 389–408, doi:10.1007/978-3-030-99253-8_20.

[16] Richard Kennaway, Jan Willem Klop, Ronan Sleep & Fer-Jan de Vries (1997): *Infinitary lambda calculus*. *Theoretical Computer Science* 175(1), pp. 93–125, doi:10.1016/S0304-3975(96)00171-5.

[17] Joachim Kock (2009): *Notes on Polynomial Functors*. Available at `https://mat.uab.cat/~kock/cat/polynomial.pdf`.

[18] Alexander Kurz, Daniela Petrişan, Paula Severi & Fer-Jan de Vries (2012): *An Alpha-Corecursion Principle for the Infinitary Lambda Calculus*. In: *CMCS* 2012, Springer, pp. 130–149, doi:10.1007/978-3-642-32784-1_8.

[19] Alexander Kurz, Daniela Petrişan, Paula Severi & Fer-Jan de Vries (2013): *Nominal Coalgebraic Data Types with Applications to Lambda Calculus*. *Logical Methods in Computer Science* 9(4), doi:10.2168/lmcs-9(4:20)2013.

[20] Joachim Lambek (1968): *A Fixpoint Theorem for complete Categories*. *Mathematische Zeitschrift* 103, pp. 151–161, doi:10.1007/BF01110627. Available at `http://eudml.org/doc/170906`.

[21] Daniel J. Lehmann & Michael B. Smyth (1981): *Algebraic specification of data types: A synthetic approach*. *Mathematical Systems Theory* 14(1), pp. 97–139, doi:10.1007/bf01752392.

[22] Lawrence S. Moss (2001): *Parametric corecursion*. *Theoretical Computer Science* 260(1–2), pp. 139–163, doi:10.1016/s0304-3975(00)00126-2.

[23] François Métayer (2003): *Fixed points of functors*. Available at `https://www.irif.fr/~metayer/PDF/fix.pdf`.

[24] Hiroshi Nakano (2000): *A modality for recursion*. In: *Proceedings of the* 15*th Annual IEEE Symposium on Logic in Computer Science*, doi:10.1109/lics.2000.855774.

[25] Andrew M. Pitts (2006): *Alpha-structural recursion and induction*. *Journal of the ACM* 53(3), pp. 459–506, doi:10.1145/1147954.1147961.

[26] Andrew M. Pitts (2013): *Nominal Sets. Names and Symmetry in Computer Science*. Cambridge University Press, doi:10.1017/CBO9781139084673.

[27] Gordon Plotkin (1990): *An Illative Theory of Relations*. In: *Situation Theory and Its Applications*, 1, CSLI, pp. 133–146. Available at `http://hdl.handle.net/1842/180`.

[28] John Power (2007): *Abstract Syntax: Substitution and Binders*. *Electronic Notes in Theoretical Computer Science* 173, pp. 3–16, doi:10.1016/j.entcs.2007.02.024.

Faster Game Solving by Fixpoint Acceleration

Daniel Hausmann*
University of Liverpool, United Kingdom
hausmann@liverpool.ac.uk

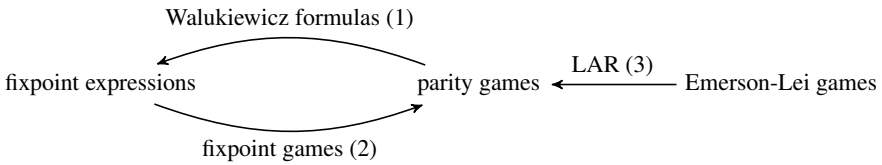
We propose a method for solving parity games with acyclic (DAG) sub-structures by computing nested fixpoints of a DAG attractor function that lives over the non-DAG parts of the game, thereby restricting the domain of the involved fixpoint operators. Intuitively, this corresponds to accelerating fixpoint computation by inlining cycle-free parts during the solution of parity games, leading to earlier convergence. We also present an economic later-appearance-record construction that takes Emerson-Lei games to parity games, and show that it preserves DAG sub-structures; it follows that the proposed method can be used also for the accelerated solution of Emerson-Lei games.

1 Background

The analysis of infinite duration games is of central importance to various problems in theoretical computer science such as formal verification (model checking), logical reasoning (satisfiability checking), or automated program construction (reactive synthesis). Previous work has shown how fixpoint expressions can be used to characterize winning in such games, which in turn has enabled the development of symbolic game solving algorithms that circumvent the state-space explosion to some extent and therefore perform reasonably well in practice in spite of the high complexity of the considered problems.

In this work, we build on the close connection between game solving and fixpoint computation to obtain a method that accelerates the solution of parity games with acyclic sub-structures. Intuitively, we base our method on the observation that cycle-free parts in parity games can be dealt with summarily by using the computation of attractors in place of handling each node individually. In games that are largely cycle-free, but in which attraction along cycle-free parts can be evaluated without exploring most of their nodes, this significantly speeds up the game solution process.

The close relation between games and fixpoint expression has been well researched. On one hand, it has been shown that winning regions in games with various objectives can be characterized by fixpoint expressions [4, 3, 11, 6]. The result for parity games (e.g. [4]) arguably is the best-known case: it has been shown that the winning region in parity games with k priorities can be specified by a modal μ-calculus formula (sometimes referred to as *Walukiewicz formulas*) with alternation-depth k; this result connects parity game solving and model checking for the μ-calculus. It corresponds to arrow (1) in the diagram below, which is meant to illustrate the context of our contribution.



A fruitful reduction in the converse direction is more recent. *Fixpoint games*, that is, parity games of a certain structure (defined in detail below), have been used to characterize the semantics of hierarchical

*Supported by the ERC Consolidator grant D-SynMA (No. 772459)

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 71–80, doi:10.4204/EPTCS.435.6

fixpoint equation systems in terms of games [2]. While this reduction (corresponding to arrow (2) in the diagram) incurs blow-up that is exponential in the size of the domain of the equation system, it still proves helpful as it allows for reasoning about nested least and greatest fixpoints in terms of (strategies in) parity games. Fixpoint games have also been used to lift the recent breakthrough quasipolynomial result for parity game solving to the evaluation of general fixpoint equation systems [9].

Later-appearance-record (LAR) constructions have been used to reduce games with richer winning objectives to parity games; this subsumes in particular the LAR-reduction for Emerson-Lei games (arrow (3)), which have recently garnered attention due to their succinctness and favorable closure properties.

In this context, the contribution of the current work is two-fold. First, we show how for parity games with cycle-free sub-structures, Walukiewicz' construction can be adapted to use multi-step attraction along cycle-free parts (we call such parts *DAGs in games*) in place of one-step attraction. In consequence, the domain of the resulting fixpoint expression can be restricted to the parts of the game that are not cycle-free, thereby accelerating convergence in the solution of the fixpoint expression. If multi-step attraction can be evaluated without visiting most cycle-free nodes (which is the case, e.g., when cycle-free parts encode predicates that can be efficiently evaluated), this trick has been shown to significantly speed up game solution. Applications can be found in both model checking [7] and satisfiability checking [8] for generalized μ-calculi in which satisfaction and joint satisfiability, respectively, of the modalities can be harder to verify than in the standard μ-calculus.

In addition to that, we propose a later-appearance-record construction for the transformation of Emerson-Lei games to equivalent parity games (corresponding to arrow (3) in the above diagram) and show that the reduction preserves cycle-free sub-structures, thereby enabling usage of the proposed acceleration method also for games with full Emerson-Lei objectives.

2 Games and Fixpoint Expressions

We start by introducing notions of games and fixpoint expressions pertaining to the acceleration method proposed in Sections 3 and 4 below.

Games. An *arena* is a graph $A = (V_\exists, V_\forall, E)$, consisting of a set $V = V_\exists \cup V_\forall$ of *nodes* and a set $E \subseteq V \times V$ of *moves*; furthermore, the set of nodes is partitioned into the sets $V_\exists$ and $V_\forall$ of nodes *owned* by player $\exists$ and by player $\forall$, respectively. We write $E(v) = \{w \in V \mid (v,w) \in E\}$ for the set of nodes reachable from node $v \in V$ by a single move. We assume without loss of generality that every node has at least one outgoing edge, that is, that $E(v) \neq \emptyset$ for all $v \in V$. A *play* $\pi = v_0 v_1 \ldots$ on A is a (finite or infinite) sequence of nodes such that $v_{i+1} \in E(v_i)$ for all $i \geq 0$. By abuse of notation we denote by A^ω the set of infinite plays on A and by A^+ the set of finite (nonempty) plays on A. A *strategy* for player $i \in \{\exists, \forall\}$ is a function $\sigma : V^* \cdot V_i \to V$ that assigns a node $\sigma(\pi v) \in V$ to every finite play πv that ends in a node $v \in V_i$. A strategy σ for player i is said to be *positional* if the prescribed moves depend only on the last nodes in plays; formally, this is the case if we have $\sigma(\pi v) = \sigma(\pi' v)$ for all $\pi, \pi' \in V^*$ and $v \in V_i$. A play $\pi = v_0 v_1 \ldots$ then is *compatible* with a strategy σ for player $i \in \{\exists, \forall\}$ if for all $j \geq 0$ such that $v_j \in V_i$, we have $v_{j+1} = \sigma(v_0 v_1 \ldots v_j)$. A strategy for player $i \in \{\exists, \forall\}$ *with memory* M is a tuple $\sigma = (M, m_0, \mathsf{update} : M \times E \to M, \mathsf{move} : V_i \times M \to V)$, where M is some set of *memory values*, $m_0 \in M$ is the initial memory value, the *update function* update assigns the outcome $\mathsf{update}(m,e) \in M$ of updating the memory value $m \in M$ according to the effects of taking the move $e \in E$, and the moving function move prescribes a single move $(v, \mathsf{move}(v,m)) \in E$ to every game node $v \in V_i$ that is owned by player i, depending on the memory value m. We extend update to finite plays π by putting $\mathsf{update}(m, \pi) = m$ in the base case that π consists of a single node, and by putting $\mathsf{update}(m, \pi) = \mathsf{update}(\mathsf{update}(m, \tau), (v,w))$ if π is of the

shape τvw, that is, contains at least two nodes. Then we obtain a strategy (without explicit memory) $\tau_\sigma : V^* \cdot V_i \to V$ from a strategy $\sigma = (M, m_0, \mathsf{update} : M \times E \to M, \mathsf{move} : V_i \times M \to V)$ with memory by putting, for all $v_0 v_1 \dots v_i \in V^* \cdot V_i$,

$$\tau_\sigma(v_0 v_1 \dots v_i) = \mathsf{move}(v_i, \mathsf{update}(m_0, v_0 v_1 \dots v_i)).$$

In this work we consider two types of objectives: *parity objectives* and, more generally, *Emerson-Lei objectives*.

Emerson-Lei objectives [10] are specified relative to a coloring function $\gamma_C : V \to 2^C$ (for some set C of colors) that assigns a set $\gamma_C(v) \subseteq C$ of colors to each node $v \in V$. A play $\pi = v_0 v_1 \dots$ then induces a sequence $\gamma_C(\pi) = \gamma_C(v_0)\gamma_C(v_1)\dots$ of sets of colors. Emerson-Lei objectives are given as Boolean formulas $\varphi_C \in \mathbb{B}(\{\mathsf{Inf}(c) \mid c \in C\})$ over atoms of the shape $\mathsf{Inf}(c)$; throughout, we write $\mathsf{Fin}(c)$ for $\neg(\mathsf{Inf}(c))$. Such formulas are interpreted over infinite sequences $\gamma_0\gamma_1\dots$ of sets of colors. We put $\gamma_0\gamma_1\dots \models \mathsf{Inf}\ c$ if and only if there are infinitely many positions i such that $c \in \gamma_i$; satisfaction of Boolean operators is defined in the usual way. Then a play π on A *satisfies* the objective φ_C if and only if $\gamma_C(\pi) \models \varphi_C$ and we define the Emerson-Lei objective induced by γ_C and φ_C by putting

$$\alpha_{\gamma_C, \varphi_C} = \{\pi \in V^\omega \mid \gamma_C(\pi) \models \varphi_C\}.$$

An *Emerson-Lei game* is a tuple $G = (A, \alpha)$, where $A = (V_\exists, V_\forall, E)$ is an arena and α is an Emerson-Lei objective. A strategy σ for player $\exists$ is *winning* at some node $v \in V$ if all infinite plays that start at v and are compatible with σ satisfy the objective α. A strategy τ for player $\forall$ is defined winning dually. The *winning region* Win_G of player $\exists$ in the game G is the set of nodes for which there is a winning strategy for player $\exists$.

As a special case, we define *parity games* to be Emerson-Lei games (A, α) where α is the Emerson-Lei objective induced by a coloring function $\gamma : V \to 2^{\{p_1, \dots, p_k\}}$ that assigns exactly one of the colors $\{p_1, \dots, p_k\}$ to each game node, and the formula

$$\bigvee_{i \text{ even}} \mathsf{Inf}(p_i) \wedge \bigwedge_{j > i} \mathsf{Fin}(p_j),$$

expressing that for some even i, color p_i is visited infinitely often and no color p_j such that j is larger than i is visited infinitely often. We denote such games by $(V_\exists, V_\forall, E, \Omega : V \to \mathbb{N})$, where Ω is a *priority function*, assigning to each node $v \in V$ the number $\Omega(v) = i$ such that $\gamma(v) = \{p_i\}$.

Emerson-Lei games are known to be *determined*, that is, every node in such games is won by exactly one of the two players. *Solving* a game then amounts to computing the winner for each game node. While the solution of Emerson-Lei games is known to be a PSPACE-complete problem [10], the precise complexity of solving parity games remains an open question; it is known however, that solving parity games is in NP∩CO-NP as well as in QP (that is, can be done deterministically in quasipolynomial time). Winning strategies in Emerson-Lei games may require exponential memory (specifically, $k!$, where k is the number of colors), but parity games have positional strategies [5].

Fixpoint Expressions. Let U be a set, k a natural number (we assume without loss of generality that k is even) and let $f : \mathscr{P}(U)^k \to \mathscr{P}(U)$ be a monotone function (such that $f(W_1, \dots, W_k) \subseteq f(W_1', \dots, W_k')$ whenever $W_i \subseteq W_i'$ for all $1 \leq i \leq k$). This data induces a fixpoint expression

$$e = \nu X_k.\, \mu X_{k-1}. \dots .\, \mu X_1.\, f(X_1, \dots, X_k).$$

The semantics of such fixpoint expressions is defined as usual, following the Knaster-Tarski fixpoint theorem. Formally, we put

$$\mathsf{LFP}\, g = \bigcap \{W \subseteq U \mid g(W) \subseteq W\} \qquad \mathsf{GFP}\, g = \bigcup \{W \subseteq U \mid W \subseteq G(W)\}$$

for monotone functions $g : \mathscr{P}(U) \to \mathscr{P}(U)$ and define

$$[\![e]\!]_i(X_{i+1},\ldots X_k) = \begin{cases} \mathsf{LFP}(\lambda X_i.[\![e]\!]_{i-1}(X_i,X_{i+1},\ldots X_k)) & \text{if } i \text{ is odd} \\ \mathsf{GFP}(\lambda X_i.[\![e]\!]_{i-1}(X_i,X_{i+1},\ldots X_k)) & \text{if } i \text{ is even} \end{cases}$$

for $1 \leq i \leq k$ and $X_{i+1},\ldots,X_k \subseteq V$, where $[\![e]\!]_0(X_1,\ldots X_k) = f(X_1,\ldots X_k)$. Then we say that $v \in V$ is *contained* in e (denoted $v \in e$) if and only if $v \in [\![e]\!]_k$; by abuse of notation, we denote $[\![e]\!]_k$ just by e.

Relation between Games and Fixpoint Expressions. The winning region in parity games can be specified by a fixpoint expression over the underlying game arena. To see how this works, let $G = (V_\exists, V_\forall, E, \Omega)$ be a parity game and define the following notation. Let $\Omega^{-1}(i) = \{v \in V \mid \Omega(v) = i\}$ denote the set of nodes that have priority i. The *controllable predecessor function* $\mathsf{Cpre} : \mathscr{P}(V) \to \mathscr{P}(V)$ that computes one-step attraction for player $\exists$ in G is defined, for $X \subseteq V$, by putting

$$\mathsf{Cpre}(X) = \{v \in V_\exists \mid E(v) \cap X \neq \emptyset\} \cup \{v \in V_\forall \mid E(v) \subset X\}.$$

Thus $\mathsf{Cpre}(X)$ contains all nodes $v \in V_\exists$ for which some outgoing move leads to X, as well as all nodes $v \in V_\forall$ for which all outgoing moves lead to X; intuitively, this is the set of nodes from which player $\exists$ can force that X is reached in one step of the game. The characterization is given by the fixpoint expression

$$\mathsf{parity}_G = \nu X_k.\, \mu X_{k-1}.\, \ldots .\, \mu X_1.\, \bigvee_{1 \leq i \leq k} \Omega^{-1}(i) \wedge \mathsf{Cpre}(X_i). \tag{1}$$

The characterization result (e.g. [4]) then is stated as follows.

Lemma 1. *We have* $\mathsf{Win}_G = \mathsf{parity}_G$.

Similar fixpoint characterizations of winning regions have been given for games with more involved objectives, including GR[1] conditions [3], Rabin and Streett objectives [11], and most recently, also for full Emerson-Lei objectives [6]. While the characterizations of winning according to these more general objectives involve fixpoint expressions of a more general format (given by hierarchical systems of fixpoint equations) than the one we have defined above, the fixpoint for the winning region in parity games will be sufficient for the purposes of this work. All mentioned fixpoint characterizations of winning regions allow for (symbolic) solution of games by computing nested fixpoints, using for instance standard fixpoint iteration [13], or using more recent techniques that employ universal trees to obtain quasipolynomial algorithms to compute nested fixpoints [9, 1].

For the converse direction, that is, going from fixpoints to games, it has recently been shown (in a more general framework than the one used in this work, see [2] for details) that the semantics of fixpoint expressions can be equivalently given in terms of parity games. The *fixpoint game* that is associated with a fixpoint expression

$$e = \nu X_k.\, \mu X_{k-1}.\, \ldots .\, \mu X_1.\, f(X_1,\ldots,X_k)$$

over U is the parity game $G_e = (V_\exists, V_\forall, E, \Omega : V \to \{1,\ldots,k\})$ played over the sets $V_\exists = U$ and $V_\forall = \mathscr{P}(U)^k \cup (U \times [k])$ of nodes; moves and priorities are given by the following table, abbreviating $(U_1,\ldots,U_k)$ by $\overline{U}$.

Node	owner	priority	allowed moves
$u \in U$	$\exists$	0	$\{\overline{U} \in \mathscr{P}(U)^k \mid u \in f(\overline{U})\}$
$\overline{U} \in \mathscr{P}(U)^k$	$\forall$	0	$\{(u,i) \in U \times [k] \mid u \in V_i\}$
$(u,i) \in U \times [k]$	$\forall$	i	$\{u\}$

When the game reaches a node $u \in U$, player $\exists$ thus has to provide a tuple $\overline{U} = (U_1, \ldots, U_k) \in \mathscr{P}(U)^k$ such that $u \in f(\overline{U})$. Player $\forall$ in turn can challenge the provided tuple by moving to any (u', i) such that $u' \in U_i$, and then continuing the game at the node $u' \in U$. Crucially, such a sequence triggers the priority i which is an even number if X_i belongs to a greatest fixpoint in e, and odd otherwise. We observe that the number of nodes in G_e is exponential in $|U|$; however, player $\exists$ owns only $|U|$ of these nodes.

Lemma 2 ([2]). *We have* $e = \mathsf{Win}_{G_e}$.

While fixpoint games enable reasoning about fixpoint expressions by reasoning about parity games, the exponential size of fixpoint games makes it unfeasible to evaluate fixpoint expressions by solving the associated fixpoint games.

3 Game Arenas with DAG Sub-structures

In this section we introduce notions related to cycle-free sub-structures in (parity) games that will be central to the upcoming acceleration result.

Definition 3. Let $G = (V_\forall, V_\exists, E)$ be a game arena. A set $W \subseteq V$ of nodes is a *DAG* (directed acyclic graph) if it does not contain an E-cycle; then there is no play of G that eventually stays within W forever. A DAG is not necessarily connected, that is, it may consist of several subgames of G. Given a DAG $W \subseteq V$, we write $V' = V \setminus W$ and refer to the set V' as *real* nodes (with respect to W). A DAG is *positional* if for each existential player node $w \in W \cap V_\exists$ in it, there is exactly one real node $v \in V'$ from which w is reachable without visiting any other real node.

We will be interested in parity games over game arenas with DAG structures, and in the computation of attraction (that is, alternating reachability) within such DAGs. To this end, we introduce the following notation of DAG attraction in parity games.

Definition 4. Let $G = (V_\forall, V_\exists, E, \Omega)$ be a parity game with priorities $\{1, \ldots, k\}$. Given a DAG W and k sets $\overline{V} = (V_1, \ldots, V_k)$ of real nodes and a real node $v \in V'$, we say that player $\exists$ can *attract* to $\overline{V}$ from v within W if there is a positional strategy σ for player $\exists$ such that for all plays π that start at v and adhere to σ, the first node v' in π such that $v' \neq v$ and $v' \in V'$ is contained in V_p, where p is the maximal priority that is visited by the part of π that leads from v to v'.

Given a DAG W, we define the *DAG attractor function* $\mathsf{DAttr}_W : \mathscr{P}(V')^k \to \mathscr{P}(V')$ by

$$\mathsf{DAttr}_W(\overline{V}) = \{v \in V \mid \text{player } \exists \text{ can attract to } \overline{V} \text{ from } v \text{ within } W\} \tag{2}$$

for $\overline{V} = (V_1, \ldots, V_k) \in \mathscr{P}(V')^k$. We assume a bound t_{Attr_W} on the runtime of computing, for any input $\overline{V} \in \mathscr{P}(V')^k$, the dag attractor of $\overline{V}$ through W.

We always have $t_{\mathsf{Attr}_W} \leq |E|$, using a least fixpoint computation to check for alternating reachability, thereby possibly exploring all DAG edges of the game. The method that we propose in the next section can play out its strength best in parity games that have large DAGs that can be efficiently evaluated (more formally, this typically means that we have both $|V'| < \log |V|$ and $t_{\mathsf{Attr}_W} < \log |V|$).

4 Large-step Solving for Parity Games

In this section we show that a parity game with DAG sub-structure W can be solved by computing a fixpoint of the DAG attractor function, using $V' = V \setminus W$ as domain of the fixpoint computation, rather than V.

So let $G = (V_\exists, V_\forall, E, \Omega)$ be a parity game with (positional) DAG W. Recall (from Lemma 1) that the winning region in G is given by

$$\mathsf{Win}_G = \nu X_k.\, \mu X_{k-1}.\, \ldots .\, \mu X_1.\, \bigvee_{1 \le i \le k} \Omega^{-1}(i) \wedge \mathsf{Cpre}(X_i),$$

where Cpre operates on subsets X_i of V. We show that

$$\mathsf{Win}_G \cap V' = \nu Y_k.\, \mu Y_{k-1}.\, \ldots .\, \mu Y_1.\, \mathsf{DAttr}_W(Y_1, \ldots, Y_k), \tag{3}$$

pointing out that $\mathsf{DAttr}_W(Y_1, \ldots, Y_k)$ is a function over subsets $Y_1, \ldots Y_k$ of V'. Therefore Equation (3) shows how attention can be reduced to non-DAG nodes (from V') when solving G; however, the price for this is that the function DAttr_W of which the fixpoint is computed is complex since it computes multi-step attraction within a DAG in place of one-step attraction as encoded by Cpre.

We make this intuition precise and prove the main result (that is, Equation (3)) as follows.

Lemma 5. *Let G be a parity game with priorities 1 to k and set V of nodes, let W be a positional DAG in G, and let $V' = V \setminus W$ and $m := |V'|$. Then $\mathsf{Win}_G \cap V'$ can be computed with $\mathscr{O}(m^{\log k})$ computations of a DAG attractor; if $k < \log m$, then $\mathsf{Win}_G \cap V'$ can be computed with a number of DAG attractor computations that is polynomial in m.*

Proof. We show that the winning regions in G can be computed by a fixpoint expression (with alternation-depth k) of the DAG attractor function given in Equation (2). Without loss of generality, assume that k is an even number.

We define the fixpoint expression

$$d = \nu X_k.\, \mu X_{k-1}.\, \ldots .\, \mu X_1.\, \mathsf{DAttr}_W(X_1, \ldots, X_k),$$

noting that the fixpoint variables X_i in d range over subsets of V' rather than over subsets of V. It has been shown in [9] how universal trees can be used to compute the nested fixpoint d with $\mathscr{O}(m^{\log k})$ (that is, quasipolynomially many) iterations of the function DAttr_W. It follows from the results of the same paper that if $k < \log m$, then d can be computed with a polynomial (in m) number of computations of a dag attractor. It remains to show that d is the winning region (restricted to V') of player $\exists$ in G.

$\Rightarrow$ Let s be a strategy with which player $\exists$ wins from all nodes in their winning region in G_d. We define a positional strategy t for player $\exists$ in G. The definition proceeds in steps that lead from a real node to the next real nodes. Given a real node $v \in V'$ that is contained in the winning region of player $\exists$ in G_d, we have $s(v) = \overline{V} \in \mathscr{P}(V')^k$ such that player $\exists$ can attract to $\overline{V}$ from v within W. For the dag part of G that starts at v, define the strategy t to simply use the positional strategy that attracts to $\overline{V}$ from v within W.

To see that t is a winning strategy, let π be a play of G that is compatible with t. Then π induces a play τ of G_d that is compatible with s: For any two positions i and i' such that $i < i'$, both $\pi(i)$ and $\pi(i')$ are real nodes and there is no $i < j < i'$ such that $\pi(j)$ is a real node, let p be the maximal priority that is visited by the partial play from $\pi(i)$ to $\pi(i')$. By construction we have $s(\pi(i)) = (V_1, \ldots, V_p, \ldots, V_k)$ where V_p contains $\pi(i')$. Thus the partial play of G that leads from $\pi(i)$ to $\pi(i')$ induces the partial play $\tau_i = \pi(i)s(\pi(i))(\pi(i'), p)\pi(i')$ of G_d. We define τ to be the concatenation of all partial plays τ_i. By construction, τ is compatible with s; as s is a winning strategy, player $\exists$ wins τ. Furthermore, the maximal priorities that are visited infinitely often in τ and π agree. Thus player $\exists$ wins π, as required.

$\Leftarrow$ Let t be a positional strategy for player $\exists$ in G with which they win from all nodes in their winning region. We define a positional strategy s for player $\exists$ in G_d as follows. For each node $v \in V'$ that is contained in the winning region of player $\exists$ in G, t yields a positional strategy that attracts from v to some $\overline{V} = (V_0, \ldots, V_k) \in \mathscr{P}(V')^k$ within W: For all v' and all $0 \leq p \leq k$ such that there is a partial play of G that is compatible with t, starts at v, ends at v', and in between visits only nodes from W, and in which the maximal priority that is visited is p, add v' to V_p. Put $s(v) = \overline{V}$. As v attracts from v to $\overline{V}$ within W by construction, s is a valid strategy.

To see that s is a winning strategy, let τ be a play of G_d that starts at a real node from the winning region of player $\exists$ in G and that is compatible with s. For each partial sub-play $v_i s(v_i)(v_{i+1}, p_i)v_{i+1}$ of τ, we define π_i to be some partial play of G that is compatible with t, starts at v_i, ends at v_{i+1}, and in between visits only nodes from W, and in which the maximal priority that is visited is p_i. Such a play exists by construction of s. Let π denote the concatenation of all π_i. Again, the maximal priorities that are visited infinitely often in τ and π agree. As t is a winning strategy, player $\exists$ wins both τ and π.

□

In case that $m < \log n$ and $t_{\mathsf{Attr}_W} \in \mathscr{O}(\log n)$ (that is, DAG attractability can be decided without exploring most of the DAG nodes), this trick leads to exponentially faster game solving.

5 Large-step Solving for Emerson-Lei Games

In this section, we show how games with an Emerson-Lei objective can be transformed to equivalent parity games, using an economic variant of the later-appearance-record (LAR) construction; we also show that this transformation preserves dag sub-structures. Thereby we enable usage of the acceleration method for parity games from the previous section also for Emerson-Lei games.

The LAR reduction annotates game nodes with permutations that act as a memory and record the order in which colors have been recently visited; this allows to identify the set of colors that is visited infinitely often. A parity condition is used to check whether this set satisfies the Emerson-Lei objective. A slight difference to previous LAR constructions for Emerson-Lei automata and games [12, 10] is that our reduction moves at most one color within the LAR in each game step, thereby allowing to bound the branching of memory updates along DAG parts of games.

Before we present the formal reduction, we first fix a set C of colors and introduce notation for permutations over C. We let $\Pi(C)$ denote the set of permutations over C, and for a permutation $\pi \in \Pi(C)$ and a position $1 \leq i \leq |C|$, we let $\pi(i) \in C$ denote the element at the i-th position of π. Let π_0 be some fixed element of $\Pi(C)$. For $D \subseteq C$ and $\pi \in \Pi(C)$, we let $\pi@D$ denote the permutation that is obtained from π by moving *the* element of D that occurs at the right-most position in π to the front of π; for instance, for $C = \{a,b,c,d\}$ and $\pi = (a,d,c,b) \in \Pi(C)$, we have $\pi@\{a,d\} = \pi@\{d\} = (d,a,c,b)$ and $(d,a,c,b)@\{a,d\} = \pi$. Crucially, restricting the reordering to single colors, rather than sets of colors, ensures that for each $\pi \in \Pi(C)$, there are only $|C|$ many π' such that $\pi@D = \pi'$ for some $D \subseteq C$. Therefore, the branching in the reduced game is bounded in such a way that a DAG with o exit points and in which every path from the entry point to some exit point visits at most one node with non-empty set of colors, is reduced to a DAG that again has o exit points. Given a permutation $\pi \in \Pi(C)$ and an index $1 \leq i \leq |C|$, we furthermore let $\pi[i]$ denote the set of colors that occur in one of the first i positions in π.

Definition 6. Let $G = (V_\exists, V_\forall, E, \gamma_C, \varphi_C)$ be an Emerson-Lei game with set C of colors. We define the parity game

$$P(G) = (V_\exists \times \Pi(C), V_\forall \times \Pi(C), E', \Omega)$$

by putting $E'(v,\pi) = \{(w, \pi@\gamma_C(v)) \mid (v,w) \in E\}$ for $(v,\pi) \in V \times \Pi(C)$, and

$$\Omega(v,\pi) = \begin{cases} 2p & \pi[p] \models \varphi_C \\ 2p+1 & \pi[p] \not\models \varphi_C \end{cases}$$

for $(v,\pi) \in V \times \Pi(C)$. Here, p denotes the right-most position in π that contains some color from $\gamma_C(v)$.

Lemma 7. *For all $v \in V$, we have $v \in \mathsf{Win}_G$ if and only if $(v,\pi_0) \in \mathsf{Win}_{P(G)}$.*

Proof. For a finite play $\tau = (v_0,\pi_0)(v_1,\pi_1)\dots(v_n,\pi_n)$ of $P(G)$, we let $p(\tau)$ denote the right-most position in π_0 such that $\pi_0(p(\tau)) \in \gamma_C(v_0 v_1 \dots v_n)$. By slight abuse of notation, we let $\Omega(\tau)$ denote the maximal Ω-priority that is visited in τ, having $\Omega(\tau) = 2(p(\tau))$ if $\pi_0[p(\tau)] \models \varphi_C$ and $\Omega(\tau) = 2(p(\tau))+1$ if $\pi_0[p(\tau)] \not\models \varphi_C$.

$\Rightarrow$ Let $\sigma = (\Pi(C), \mathsf{update}_\sigma, \mathsf{move}_\sigma)$ be a strategy with memory $\Pi(C)$ for player $\exists$ in G with which they win every node from their winning region. It has been shown in a previous LAR reduction for Emerson-Lei games [10] that winning strategies with this amount of memory always exist. We define a strategy $\rho = (\Pi(C), \mathsf{update}_\rho, \mathsf{move}_\rho)$ with memory $\Pi(C)$ for player $\exists$ in $P(G)$ by putting $\mathsf{update}_\rho(\pi, ((v,\pi'),(w,\pi''))) = \mathsf{update}_\sigma(\pi,(v,w))$ and

$$\mathsf{move}_\rho((v,\pi'),\pi) = (\mathsf{move}_\rho(v,\pi), \pi@\gamma_C(v)).$$

Thus ρ updates the memory and picks moves just as σ does, but also updates the permutation component in $P(G)$ according to the taken moves; hence ρ is a valid strategy.

We show that ρ wins a node (v,π) in $P(G)$ whenever v is in the winning region of player $\exists$ in G. To this end, let $\tau = (v_0,\pi_0)(v_1,\pi_1)\dots$ be a play of $P(G)$ that starts at $(v_0,\pi_0) = (v,\pi)$ and is compatible with ρ. By construction, $\theta = v_0 v_1 \dots$ is a play that is compatible with σ. Since σ is a winning strategy for player $\exists$, we have $\gamma_C(\theta) \models \varphi_C$. There is a number i such that all colors that appear in θ from position i on occur infinitely often. Let p be the number of colors that appear infinitely often in θ. It follows by definition of $\pi@D$ for $D \subseteq C$ (which moves the single right-most element of π that is contained in D to the very front of π), that there is a position $j \geq i$ such that the left-most p elements of π_j are exactly the colors occurring infinitely often in π (and all colors to the right of $\pi_j(p)$ are never visited from position j on). It follows that from position j on, τ never visits a priority larger than $2p$. To see that τ infinitely often visits priority $2p$ we note that $\pi_{j'}[p] \models \varphi_C$ for any $j' > j$, so it suffices to show that p infinitely often is the rightmost position in the permutation component of τ that is visited. This is the case since, from position j on, the p-th element in the permutation component of τ cycles fairly through all colors that are visited infinitely often by θ.

$\Leftarrow$ Let ρ be a positional strategy for player $\exists$ in $P(G)$ with which they win every node from their winning region. We define a strategy $\sigma = (\Pi(C), \mathsf{update}_\sigma, \mathsf{move}_\sigma)$ with memory $\Pi(C)$ for player $\exists$ in G by putting $\mathsf{update}_\sigma(\pi,(v,w)) = \pi@\gamma_C(v)$ and $\mathsf{move}_\sigma(v,\pi) = w$ where w is such that $\rho(v,\pi) = (w, \pi@\gamma_C(v))$. Thus σ updates the memory and picks moves just as plays that follow ρ do.

We show that σ wins a node v in G whenever (v,π) is in the winning region of player $\exists$ in $P(G)$. To this end, let $\theta = v_0 v_1 \dots$ be a play of G that starts at $v_0 = v$ and is compatible with σ. By construction, θ induces a play $\tau = (v_0,\pi_0)(v_1,\pi_1)\dots$ with $(v_0,\pi_0) = (v,\pi)$ and $\pi_{i+1} = \pi_i @ \gamma_C(v_i)$ for $i \geq 0$ that is compatible with ρ. Since ρ is a winning strategy for player $\exists$, the maximal priority in it is even (say $2p$). Again, p is the position such that the left-most p elements in the permutation component of ρ from some point on contain exactly the colors that are visited infinitely often by θ. It follows that $\gamma_C(\theta) \models \varphi_C$.

□

The LAR reduction from Definition 6 preserves DAG sub-structures:

Lemma 8. *Let G be an Emerson-Lei games with set C of colors. If W is a positional DAG in G, then $W \times \Pi(C)$ is a positional DAG in $P(G)$.*

Proof. The claim follows immediately since the LAR reduction just annotates game nodes with additional memory, meaning that each game node v in G is replaced with copies (v,π) in $P(G)$ so that all cycles in $P(G)$ have corresponding cycles in G. □

6 Conclusion

We have presented a method for solving parity games with DAG sub-structures by computing nested fixpoints over the non-DAG nodes only, thereby intuitively accelerating fixpoint computation by summarizing cycle-free parts during the solution of parity games. This can significantly reduce the domain of the game solving process, and in cases where DAG sub-structures in addition can be evaluated without exploring all nodes in them, it improves the time complexity of parity game solving. Furthermore, we have proposed a later-appearance-record construction with linear branching on the memory values that transforms Emerson-Lei games to parity games, and have shown that this transformation preserves DAG sub-structures, enabling usage of the proposed method also for the solution of games with general Emerson-Lei objectives.

References

[1] André Arnold, Damian Niwinski, and Pawel Parys. A quasi-polynomial black-box algorithm for fixed point evaluation. In *Computer Science Logic, CSL 2021*, volume 183 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CSL.2021.9.

[2] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games on continuous lattices. *Proc. ACM Program. Lang.*, 3(POPL):26:1–26:29, 2019. doi:10.1145/3290339.

[3] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012. doi:10.1016/j.jcss.2011.08.007.

[4] Florian Bruse, Michael Falk, and Martin Lange. The fixpoint-iteration algorithm for parity games. In *Games, Automata, Logics and Formal Verification, GandALF 2014*, volume 161 of *EPTCS*, pages 116–130, 2014. doi:10.4204/EPTCS.161.12.

[5] Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Logic in Computer Science*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

[6] Daniel Hausmann, Mathieu Lehaut, and Nir Piterman. Symbolic solution of Emerson-Lei games for reactive synthesis. In *Foundations of Software Science and Computation Structures, FoSSaCS 2024*, volume 14574 of *LNCS*, pages 55–78. Springer, 2024. doi:10.1007/978-3-031-57228-9_4.

[7] Daniel Hausmann and Lutz Schröder. Game-based local model checking for the coalgebraic μ-calculus. In *Concurrency Theory, CONCUR 2019*, volume 140 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.CONCUR.2019.35.

[8] Daniel Hausmann and Lutz Schröder. Optimal satisfiability checking for arithmetic μ-calculi. In *Foundations of Software Science and Computation Structures, FOSSACS 2019*, volume 11425 of *LNCS*, pages 277–294. Springer, 2019. doi:10.1007/978-3-030-17127-8_16.

[9] Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021*, volume 12651 of *LNCS*, pages 38–56. Springer, 2021. doi:10.1007/978-3-030-72016-2_3.

[10] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *Mathematical Foundations of Computer Science*, volume 3618 of *LNCS*, pages 495–506. Springer, 2005. doi:10.1007/11549345_43.

[11] Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. In *Logic in Computer Science, LICS 2006*, pages 275–284. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.23.

[12] Florian Renkin, Alexandre Duret-Lutz, and Adrien Pommellet. Practical "paritizing" of Emerson-Lei automata. In *Automated Technology for Verification and Analysis, ATVA 2020*, volume 12302 of *LNCS*, pages 127–143. Springer, 2020. doi:10.1007/978-3-030-59152-6_7.

[13] Helmut Seidl. Fast and simple nested fixpoints. *Universität Trier, Mathematik/Informatik, Forschungsbericht*, 96-05, 1996.

Characterizing the Exponential-Space Hierarchy Via Partial Fixpoints

Florian Bruse David Kronenberger Martin Lange
University of Kassel, Germany
{florian.bruse|martin.lange}@uni-kassel.de

The characterization of PSPACE-queries over ordered structures as exactly those expressible in first-order logic with partial fixpoints (Vardi'82) is one of the classical results in the field of descriptive complexity. In this paper, we extend this result to characterizations of k-EXPSPACE-queries for arbitrary k, characterizing them as exactly those expressible in order-$k+1$-higher-order logic with partial fixpoints. For $k > 1$, the restriction to ordered structures is no longer necessary due to the high expressive power of higher-order logic.

1 Introduction

Computational complexity studies the difficulty of computation problems with regards to the consumption of computational resources, most prominently time and space. Descriptive complexity, as a subdomain of both computational complexity and formal logic, has taken this study to a more abstract level by characterizing classes of problems, i.e., complexity classes, through logical definability. This achieves the characterization of the difficulty, resp. complexity of problems without resorting to measuring the use of computational resources, as this ultimately depends on the choice of an underlying model of computation like a Turing machine for instance. Descriptive complexity thus manages to characterize the difficulty of problems through the structure of the problem alone, regardless of an underlying model of computation. One can argue, though, that the resources used to measure complexity are logical operators that give the underlying logics their expressiveness, like predicate or fixpoint quantifiers.

Descriptive complexity started off with Fagin's seminal result [3] showing that the well-known complexity class NP coincides with $\exists$SO, the set of problems definable in existential second-order logic. Stockmeyer extended this to a characterization of problems between NP and PSPACE by means of second-order logic (SO), known as the polynomial hierarchy (PH) [9].

An interesting – and still open – question asks for a logical characterization of the complexity class P. This is believed to open ways to tackle the famous P=NP question. One of the major obstacles here is the lack of a total order on the elements of a structure forming an instance of some computational problem, like a graph for instance. When processing graphs with a computational model like a Turing machine, it can be assumed to be totally ordered due to the way that it needs to be represented as an input. For logical formulas, operating directly on structures and not on string representations thereof, this is not the case. On the other hand, a total order helps immensely; it enables iteration over all elements of the structure. Moreover, a logical characterization of the complexity class P is known when inputs to its problems are assumed to be explicitly ordered. This is known as the Immerman-Vardi Theorem [5, 10], stating that the complexity class P on ordered structures is captured by the extension of first-order logic with least fixpoint quantifiers (FO+LFP).

Fixpoint quantifiers turned out to be a useful tool in descriptive complexity. Immerman lifted the Immerman-Vardi Theorem to a characterization of the complexity class EXPTIME by second-order

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 81–91, doi:10.4204/EPTCS.435.7

logic with least fixpoint quantifiers (SO+LFP) [6]. Note that the fixpoint quantifiers in SO+LFP are not the same as the ones in FO+LFP. The LFP in FO+LFP refers to least fixpoints of first-order functions mapping tuples of elements to tuples of elements. This can be expressed in SO, i.e. FO+LPF $\subseteq$ SO. The LFP in SO+LFP refers to fixpoints of second-order functions, mapping predicates to predicates. This naturally gives rise to the question after characterizations of classes in the exponential-time hierarchy by means of higher-order logic with fixpoints. Indeed, Freire and Martins [4] showed that for any $k \geq 2$, the class k-EXPTIME of problems solvable in k-fold exponential time is captured by HO^{k+1}+LFP, i.e. higher-order formulas of order at most $k+1$ with corresponding least fixpoint quantifiers.

Given the rather complete picture for time complexity, it is natural to ask whether space complexity is also open to logical characterizations in the same fashion. Another celebrated result in descriptive complexity, made use of in e.g., the Abiteboul-Vianu Theorem [1], is due to Vardi [10] (not to be confused with the Immerman-Vardi Theorem, from [5] and also [10]). It states that the class PSPACE on ordered structures is captured by FO+PFP, i.e., the extension of first-order logic by *partial* fixpoints.

In this paper we extend the descriptive complexity of classes in the exponential space hierarchy with the Vardi's result at the basis, just like Freire and Martins have done for the time hierarchy with the Immerman-Vardi Theorem at its basis. We show that, for any $k \geq 1$, the complexity class k-EXPSPACE of problems solvable using at most k-fold exponential space, is captured by the logic HO^{k+1}+PFP of formulas of order at most $k+1$ with partial fixpoint quantifiers.

2 Preliminaries

Let $n,k \in \mathbb{N}$. We write 2^n_k for the following: $2^n_k = n$ if $k = 0$, and $2^n_{k+1} = 2^{2^n_k}$.

2.1 Space-bounded Turing Machines

A deterministic Turing machine (DTM) is a tuple $\mathscr{M} = (Q, \Sigma, \Gamma, \Box, \delta, q_{\text{init}}, q_{\text{acc}}, q_{\text{rej}})$ where Q is a finite set of *states*, Σ is a finite, nonempty *input alphabet*, $\Gamma \supseteq \Sigma$ is a finite, nonempty *tape alphabet*, $\Box \in \Gamma \setminus \Sigma$ is the *blank symbol*, $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L,N,R\}$ is the *transition function*, and $q_{\text{init}}, q_{\text{acc}}, q_{\text{rej}} \in Q$ are the unique *starting*, *accepting* and *rejecting states*.

A *configuration* of a DTM is a tuple (q,h,t) where $q \in Q$ is the current state, $h \in \mathbb{N}$ is the head position, and $t \colon \mathbb{N} \to \Gamma$ is the tape content. The *initial configuration* on input word $w \in \Sigma^*$ is given by $(q_{\text{init}}, 0, t)$ with $t(i) = w_i$ if $i < |w|$ and $t(i) = \Box$ otherwise. The unique *accepting* and *rejecting configurations* are given by $(q_{\text{acc}}, 0, t)$, resp. $(q_{\text{rej}}, 0, t)$ where $t(i) = \Box$ for all i in both cases.

A configuration $C = (q', h', t')$ is the, necessarily unique, *successor configuration* of (q,h,t) if (I) $q \notin \{q_{\text{acc}}, q_{\text{rej}}\}$, (II) $\delta(q, t(h)) = (q', \gamma, D)$ for some $\gamma \in \Gamma, D \in \{L,N,R\}$, (III) $t'(i) = \gamma$ if $i = h$ and $t'(i) = t(i)$ otherwise, and (IV) $h' = h-1$ if $D = L$ and $h > 0$, $h' = h$ if $D = N$, and $h' = h+1$ if $D = R$. A (partial) *computation* of $\mathscr{M}$ on input w is a finite or infinite sequence of configurations $C_0, C_1, \ldots$ where C_0 is the initial configuration of $\mathscr{M}$ on w, and C_{i+1}, if it exists, is the successor configuration of C_i. Such a computation is *maximal* if it is either infinite or its last configuration is the accepting or the rejecting configuration. Note that each $\mathscr{M}$ has exactly one maximal computation for each w, whence from now on we talk about *the* computation of $\mathscr{M}$ on w. We say that $\mathscr{M}$ *accepts* w if its unique maximal computation on w ends with the accepting configuration, and we write $L(\mathscr{M})$ for the set of words accepted by $\mathscr{M}$. Conversely, $\mathscr{M}$ *rejects* w if its unique maximal computation on w ends in the rejecting configuration or if it is infinite. In the latter case, we say that the computation *diverges*.

We say that a non-diverging computation $C_0, \ldots, C_k$ on input some w consumes space n, written

$\text{space}_{\mathscr{M}}(w) = n$, if $n = 1 + \max\{h_i \mid C_i = (q_i, h_i, t_i)\}$. Obviously, if the head never advances beyond position $n-1$, then $t_i(j) = \square$ for all $0 \leq i \leq k$ and $j > n-1$. Let $f\colon \mathbb{N} \to \mathbb{N}$ be a function. We say that $\mathscr{M}$ is f-space-bounded if $\mathscr{M}$ has no diverging computations on any input and, for all n, we have $\max\{\text{space}_{\mathscr{M}}(w) \mid w \in \Sigma^n\} \leq f(n)$. We say that $\mathscr{M}$ is *k-fold-exponential space bounded* if there is a polynomial $p(n)$ such that $\mathscr{M}$ is $2_k^{p(n)}$-space-bounded.

2.2 Higher-Order Logic with Partial Fixpoints

In order to keeps things notationally simple, we restrict ourselves to the class of labeled transition systems (LTS), or labelled graphs. Let $\mathbf{P} = \{p, q, \dots\}$ be a set of *propositions* and let $\mathbf{A} = \{a, b, \dots\}$ be a set of *actions* or *transition relation* or *edge relations*. An LTS is a tuple $T = (S, A, \ell)$ where $S = \{s, t, \dots\}$ is a finite, nonempty set of *states*, $A \subseteq S \times \mathbf{A} \times S$ is the *transition relation* and $\ell\colon S \to 2^{\mathbf{P}}$ labels each state by the set of propositions valid in it. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in A$.

Types. The set of types is defined via the grammar

$$\tau, \tau_1, \dots, \tau_n ::= \bullet \mid \tau_1, \dots, \tau_n \mid (\tau)$$

where $\bullet$ is the *ground type* or *type of individuals* of *order* $ord(\bullet) = 1$, $\tau_1, \dots, \tau_n$ is a *compound type* of order $ord(\tau_1, \dots, \tau_n) = \max\{ord(\tau_1), \dots, ord(\tau_n)\}$, and where (τ) is a *set type* of order $ord((\tau)) = 1 + ord(\tau)$.[1]

Given an LTS $T = (S, A, \ell)$, the semantics $[\![\tau]\!]^T$ of a type τ is given by

$$[\![\bullet]\!]^T = S, \qquad [\![\tau_1, \dots, \tau_n]\!]^T = [\![\tau_1]\!]^T \times \cdots \times [\![\tau_n]\!]^T \qquad [\![(\tau)]\!]^T = 2^{[\![\tau]\!]^T}.$$

We often compress compound and set types by writing e.g., $(\tau_1, \dots, \tau_n)$.

The following is straightforward to prove by induction on the structure of types.

Lemma 1. *For any τ of order k and any LTS T, with state set S, $|[\![\tau]\!]^T|$ is $k-1$-fold exponential in $|S|$.*

Given an LTS as above, and some $f\colon [\![\tau]\!]^T \to [\![\tau]\!]^T$ for τ of order at least 2, we define its *partial fixpoint* PFP f via

$$\text{PFP } f = \begin{cases} F^i, \text{ if } i \text{ exists such that } F^i = F^{i+1} \\ \emptyset, \text{ otherwise,} \end{cases}$$

where $F^0 = \emptyset$ and $F^{i+1} = f(F^i)$. By an obvious counting argument, if there is i such that a nontrivial partial fixpoint exists, then there already is one bounded by $|[\![\tau]\!]^T|$, which, by Lem. 1, is $k-1$-fold exponential in $|S|$ for τ of order k.

Syntax. Let $\mathscr{V} = \{X, Y, \dots\}$ be a set of typed variables, tacitly assumed to contain infinitely many variables for each type. The set of HO+PFP-formulas is defined by the grammar

$$\varphi ::= \mathtt{tt} \mid p(X) \mid a(X, Y) \mid X(Y_1, \dots, Y_n) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists(X\colon \tau).\ \varphi \mid (\text{PFP}(X\colon \tau).\ \varphi)(Y_1, \dots, Y_n)$$

where $p \in \mathbf{P}$, $a \in \mathbf{A}$, and $X, Y_1, \dots, Y_n$, are variables. A formula φ is *well-formed* if the following are true for φ: (I) The variables in terms of the form $p(X)$ or $a(X, Y)$ are of type $\bullet$, and (II) in a term of the

[1] Compound type and set type are often combined into a single "set of tuples" type. We use separate operators here for ease of notation, but the results of the paper do not depend on that.

form $X(Y_1,\ldots,Y_n)$ or $(\mathrm{PFP}(X\colon \tau).\ \varphi)(Y_1,\ldots,Y_n)$, the variable X has type $(\tau_1,\ldots,\tau_n)$ if Y_i has type τ_i for $1 \le i \le n$. If they are not important, we omit type annotations of the form $(X\colon \tau)$, and we use compressed notation such as $\exists(X,Y,Z\colon \bullet).\ \varphi$ or $\exists(X,Y,Z\colon \tau,\tau',\tau'')$ where appropriate.

Other derived operators such as $\wedge,\rightarrow,\forall,\mathtt{ff}$ etc. can be added in the usual way. The notions of subformula, formula size etc. are also standard. Free and bound variables are defined as usual, with X being a bound variable in $(\mathrm{PFP}(X\colon \tau).\ \varphi)(Y_1,\ldots,Y_n)$. We use notation such as $\varphi(X\colon \tau, Y\colon \tau')$ etc. to communicate the names and types of the free variables of a formula, with shorthands as above used if appropriate.

We say that φ has order k if the highest order of a variable that occurs freely or as X in a formula of the form $\exists X.\ \psi$ is at most k, and the highest order of a variable X in a subformula of the form $(\mathrm{PFP}(X\colon \tau).\ \varphi)(Y_1,\ldots,Y_n)$ is at most $k+1$. We write $\mathsf{HO}^k\text{+}\mathsf{PFP}$ for the collection of all formulas of order at most k.

Semantics. Let $T=(S,A,\ell)$ be an LTS. A variable assignment η is a function that assigns, to each variable $X \in \mathscr{V}$ of type τ, an element of $[\![\tau]\!]^T$. Given some X of type τ and some $f \in [\![\tau]\!]^T$, the update $\eta[X \mapsto f]$ is defined as $\eta[X \mapsto f](X) = f$ and $\eta[X \mapsto f](Y) = \eta(Y)$ if $Y \neq X$.

The semantics of a HO+PFP formula is defined as follows:

$$\begin{aligned}
T,\eta &\models \mathtt{tt}, \text{ always}\\
T,\eta &\models p(X), \text{ iff } p \in \ell(\eta(x))\\
T,\eta &\models a(X,Y), \text{ iff } \eta(X) \xrightarrow{a} \eta(Y)\\
T,\eta &\models X(Y_1,\ldots,Y_n), \text{ iff } (\eta(Y_1),\ldots,\eta(Y_n)) \in \eta(X)\\
T,\eta &\models \neg\varphi, \text{ iff } T,\eta \not\models \varphi\\
T,\eta &\models \varphi_1 \vee \varphi_2, \text{ iff } T,\eta \models \varphi_1 \text{ or } T,\eta \models \varphi_2\\
T,\eta &\models \exists(X\colon \tau).\varphi, \text{ iff there is } f \in [\![\tau]\!]^T \text{ s.t. } T,\eta[X \mapsto f] \models \varphi\\
T,\eta &\models (\mathrm{PFP}(X\colon \tau)\varphi)(Y_1,\ldots,Y_n), \text{ iff } (\eta(Y_1),\ldots,\eta(Y_n)) \in \mathrm{PFP}\ \varphi^T_\eta
\end{aligned}$$

where φ^T_η is the function that maps $g \in [\![\tau]\!]^T$ to

$$\{(g_1,\ldots,g_n) \in 2^{[\![\tau_1]\!]^T \times\cdots\times [\![\tau_n]\!]^T} \mid T,\eta[X \mapsto g, Y_1 \mapsto g_1,\ldots,Y_n \mapsto g_n] \models \varphi\} \in [\![\tau]\!]^T$$

if $\tau = (\tau_1,\ldots,\tau_n)$.

2.3 Queries

Let $\mathbf{P}$ and $\mathbf{A}$ be fixed. A (boolean) *query* (over $\mathbf{P}$ and $\mathbf{A}$) is a function $\mathscr{Q}$ that maps, to each finite LTS $T=(S,A,\ell)$ a truth value, i.e., either true or false. Alternatively, such a boolean query is just a set of finite LTS over (over $\mathbf{P}$ and $\mathbf{A}$), which we shall identify with $\mathscr{Q}$.

A closed $\mathsf{HO}\text{+}\mathsf{PFP}$ formula φ naturally defines a query via

$$\mathscr{Q}_\varphi = \{T \mid T \models \varphi\}.$$

Conversely, queries can be decided by space-bounded Turing machines. For this, the machine receives the LTS in question as an input, and either accepts or rejects. The LTS has to be encoded into some word of the input alphabet for this. Naturally, this introduces a total order on the set of states of the

LTS. It is known that e.g., the expressive power of first-order logic increases in the presence of an order (this is a classic exercise when introducing Ehrenfeucht-Fraïssé games). However, order is not an issue in our setting. The classical first-order characterization due to Abiteboul and Vianu is explicitly restricted to ordered structures, and characterizations for logics beyond existential second-order logic can be done with an order in mind, as existential second-order logic is strong enough to simply guess an order. This includes HO^k+PFP for $k \geq 2$, i.e., the topic of this paper.

Given an LTS T, let $\langle T \rangle$ be some form of polynomial encoding of T into a given input alphabet Σ, e.g., using adjacency matrices or the like. We say that a Turing machine $\mathscr{M}$ decides a query $\mathscr{Q}$ if $\mathscr{M}$ halts on any input of the form $\langle T \rangle$, where T is necessarily finite, and accepts exactly those codings where $T \in \mathscr{Q}$. A query is a k-EXPSPACE-query if there is $\mathscr{M}$ that is k-fold-exponential space bounded and decides $\mathscr{Q}$.

We now say that a logic $\mathscr{L}$ captures a complexity class $\mathscr{C}$ over a class of structures (LTS) $\mathscr{S}$ if, for each $\mathscr{L}$-query there is a $\mathscr{C}$-query that yields the same set when restricted to $\mathscr{S}$, and vice versa.

Remark 2. Non-boolean queries are quite common in e.g., the field of database theory. A d-query is then not a function that maps an LTS to a truth value, but rather one that maps an LTS and a d-tuple of states to a truth value, or, equivalently, maps every LTS to a set of d-tuples. On the logical side, one now deals with formulas with free first-order variables. We choose to stick to boolean queries here in order to avoid the extra coding required to get said free variables encoded into DTM.

2.4 Vardi's Characterization of PSPACE

We briefly sketch the classical result due to Vardi [10] that first-order logic with partial fixpoints, i.e., HO^1+PFP, captures PSPACE over the class of ordered LTS. One direction is rather straightforward since first-order queries can be evaluated in polynomial *time*, the individual stages of a partial fixpoint only take polynomial space, and the next stage can be computed from the previous one also in polynomial time. Since such a partial fixpoint either does not stabilize, or stabilizes after at most exponentially many iterations, it is sufficient to keep a counter for the number of iterations, which takes polynomially many bits if it is encoded in binary.

For the other direction, let $\mathscr{M} = (Q, \Sigma, \Gamma, \square, \delta, q_{\text{init}}, q_{\text{acc}}, q_{\text{rej}})$ be a $p(n)$-space-bounded DTM that decides a query $\mathscr{Q}$ over LTS, i.e., it accepts those $w = \langle T \rangle$ such that $T \in \mathscr{Q}$.

Since $\mathscr{M}$ is $p(n)$-space-bounded, the tape contents and the head position of each configuration of a computation of $\mathscr{M}$ on an input of length n can be represented by a number of at most $p(n)$ and a word of length $p(n)$ over the tape alphabet of $\mathscr{M}$. The proof rests on three key observations:

- In sufficiently large, ordered LTS, a configuration of $\mathscr{M}$ can be represented as a second-order relation of sufficient arity,
- the operator that computes from such a representation of a configuration its successor configuration, if it exists, can be expressed as a first-order formula, and
- the initial and accepting configurations can be pinned down using first-order logic.

The capturing result is then obtained by observing that $\mathscr{M}$ accepts its input w, derived from T, iff the partial fixpoint obtained by feeding a representation of the initial configuration into the operator mentioned above is nonempty and contains exactly a representation of the accepting configuration.

3 HO^k+PFP-Queries are in $k-1$-EXPSPACE

We begin with the simpler part of the capturing result. We will show that queries definable in HO^k+PFP can be evaluated using at most $(k-1)$-fold exponential space. This does not even need any special tricks. Alg. 1 essentially just computes the semantics of an HO^k+PFP query φ w.r.t. an LTS T and a variable evaluation η, i.e., it decides whether or not $T,\eta \models \varphi$ holds.

Algorithm 1 Evaluating HO^k+PFP queries in $(k-1)$-fold exponential space.

```
1:  procedure EVAL(T, η, φ)
2:      case φ of
3:      tt:  return true
4:      p(X):  return p ∈ ℓ(η(X))
5:      a(X,Y):  return η(X) -a-> η(Y)
6:      X(Y_1,...,Y_n):  return (η(Y_1),...,η(Y_n)) ∈ η(X)
7:      ¬ψ:  return ¬EVAL(T, η, ψ)
8:      ψ_1 ∨ ψ_2:  return EVAL(T, η, ψ_1) ∨ EVAL(T, η, ψ_2)
9:      ∃(X : τ). ψ:
10:         for all f ∈ [[τ]]^T do
11:             if EVAL(T, η[X ↦ f], ψ) then
12:                 return true
13:             end if
14:         end for
15:         return false
16:     (PFP(X : (τ_1,...,τ_k)). ψ)(Y_1,...,Y_n):
17:         f ← ∅
18:         cnt ← 0
19:         while cnt < |[[(τ_1,...,τ_k)]]^T| do
20:             f' ← f
21:             f ← ∅
22:             for all (M_1,...,M_k) ∈ [[τ_1]]^T × ··· × [[τ_k]]^T do
23:                 if T, η[Y_1 ↦ M_1,...,Y_k ↦ M_k, X ↦ f'] ⊨ ψ then
24:                     f ← f ∪ {(M_1,...,M_k)}
25:                 end if
26:             end for
27:             if f = f' then
28:                 return (η(Y_1),...,η(Y_n)) ∈ f
29:             end if
30:             cnt ← cnt + 1
31:         end while
32:         return false
33:     end case
34: end procedure
```

Theorem 3. *Let $k \geq 2$. Evaluating an HO^k+PFP query is in $(k-1)$-EXPSPACE.*

Proof. It is not hard to see that algorithm EVAL correctly evaluates an HO+PFP query, as it closely

follows the semantics of HO+PFP. It remains to be seen that the space needed by this procedure is bounded by a function that is at most $(k-1)$-fold exponential in the size of the underlying $|T|$ and $|\varphi|$.

First note that the recursion depth in EVAL is bounded by $|\varphi|$. Hence, it suffices to check that the space needed within each recursive call is bounded in this way. It is only the last two cases in which this may not be obvious. So consider the case of $\varphi = \exists(X\colon \tau).\ \psi$. Enumerating all elements of $[\![\tau]\!]^T$ requires space for one of these elements, plus space either for a counter to abort the enumeration after all elements have been constructed, or for a second of these elements in case the enumeration can construct, from one of these elements, a uniquely determined successor (in a lexicographic ordering for instance). In both cases, the space needed is logarithmic in $|[\![\tau]\!]^T|$ which is at most $(k-1)$-fold exponential in $|T|$ according to Lemma 1.

The argument for the last case of $\varphi = (\mathrm{PFP}(X\colon (\tau_1,\dots,\tau_k))\psi)(Y_1,\dots,Y_n)$ is similar. We write τ for $(\tau_1,\dots,\tau_k)$. Note that the order of τ may be up to $k+1$, so $|[\![\tau]\!]^T|$ is k-fold exponential in S. The space needed to evaluate the partial fixpoint formula is determined by a counter with values up to $|[\![\tau]\!]^T|$ and by the two elements $f, f' \in [\![\tau]\!]^T$. Using binary coding, the space needed for the counter is logarithmic in $|[\![\tau]\!]^T|$, and individual elements of $[\![\tau]\!]^T$ take $k-1$-fold exponential space, too. Hence, the space needed in this case is also at most $(k-1)$-fold exponential in $|T|$. □

4 $k-1$-EXPSPACE-Queries are Expressible in HO^k+PFP

4.1 Ordering Higher-Order Relations

Since we want to encode runs of k-fold-exponentially space-bounded Turing machines into formulas of HO^{k+1}+PFP, we have to be able to encode the tape contents of the Turing machine in question. For such a space-bounded machine, the tape can be represented by a Γ-word of k-fold exponential length, where Γ is the tape alphabet of the machine in question. Hence, we have to be able to somehow represent such a large word or, in other words, we must be able to count to large numbers.

Let $p(n)$ be a polynomial, for the time being one of the form n^c for some $c \geq 2$. Let $\mathbf{A}$ contain a relation $<$, and let the types $\tau_0,\dots,\tau_k$ be the types defined via $\tau_1 = \bullet^c$, i.e. $\bullet \times \cdots \times \bullet$ with c many repetitions of $\bullet$, and $\tau_{i+1} = (\tau_i)$. We define formulas $\varphi^1_<$, $\varphi^2_<$ and $\varphi^{i+1}_<$ for $i \geq 2$ via:

$$\varphi^1_<(X_1,\dots,X_c,Y_1,\dots,Y_c\colon \bullet) = \bigvee_{i=1}^{c} < (X_i,Y_i) \wedge \bigwedge_{j=1}^{i-1} \neg < (Y_j,X_j)$$

$$\varphi^2_<(X,Y\colon \tau_1) = \exists(Z_1,\dots,Z_c\colon \bullet).\, Y(Z_1,\dots,Z_c) \wedge \neg X(Z_1,\dots,Z_c) \wedge \forall(Z'_1,\dots,Z'_c\colon \bullet).\, \big(\varphi^1_<(Z'_1,\dots,Z'_c,Z_1,\dots,Z_c) \to X(Z'_1,\dots,Z'_c) \to Y(Z'_1,\dots,Z'_c)$$

$$\varphi^{i+1}_<(X,Y\colon \tau_{i+1}) = \exists(Z\colon \tau_i).\, Y(Z) \wedge \neg X(Z) \wedge \forall(Z'\colon \tau_i).\, \varphi^i_<(Z',Z)) \to \big(X(Z') \to Y(Z')\big)$$

Here, $\varphi^1_<$ defines a total order on $[\![\bullet^c]\!]^T$ via the lexicographical ordering induced by $<$. For $i \geq 1$, the formula $\varphi^{i+1}_<$ then totally orders $[\![\tau_{i+1}]\!]^T$ via lexicographical ordering of sets w.r.t. the membership of elements of τ_i.

Lemma 4. *Let $< \in \mathbf{A}$ and let $T = (S,A,\ell)$ be an LTS over $\mathbf{A}$ and some $\mathbf{P}$ such that $<$ is a total order on S. Let τ_k for $k \geq 1$ be defined as above. Then the following are true for all $k \geq 1$: (I) $|[\![\tau_k]\!]^T| = 2^{|S|}_{k-1}$, (II) $\varphi^k_<$ defines a total order on $[\![\tau_k]\!]^T$.*

Additionally, let $\varphi^1_=, \varphi^i_=$ and $\varphi^1_{\text{succ}}, \varphi^i_{\text{succ}}$ for $i > 1$ be defined as

$$\begin{aligned}
\varphi^1_=(X_1,\dots,X_c,Y_1,\dots,Y_c\colon \bullet) &= \neg\varphi^1_<(X_1,\dots,X_c,Y_1,\dots,Y_c) \wedge \neg\varphi^1_<(Y_1,\dots,Y_c,X_1,\dots,X_c)\\
\varphi^i_=(X,Y\colon \tau_i) &= \neg\varphi^i_<(X,Y) \wedge \neg\varphi^i_<(Y,X)\\
\varphi^1_{\text{succ}}(X_1,\dots,X_c,Y_1,\dots,Y_c\colon \bullet) &= \varphi^1_<((X_1,\dots,X_c,Y_1,\dots,Y_c) \wedge \forall(Z_1,\dots,Z_c\colon \bullet).\, \varphi^1_=(X_1,\dots,X_c,Y_1,\dots,Y_c)\\
&\qquad \rightarrow (\varphi^1_=(X_1,\dots,X_c,Z_1,\dots,Z_c) \vee \varphi^1_<(Z_1,\dots,Z_c,X_1,\dots,X_c))\\
\varphi^i_{\text{succ}}(X,Y\colon \tau_i) &= \varphi^i_<(X,Y) \wedge \forall(Z\colon \tau_i).\, \varphi_<(Z,Y) \rightarrow \varphi_=(X,Z) \vee \varphi_<(Z,X)
\end{aligned}$$

expressing equality between elements of $[\![\tau_i]\!]^T$ or the fact that the second argument is the immediate successor of the first one w.r.t. the total order induced by $\varphi^i_<$.

Finally, for each $j \in \mathbb{N}$ and $i > 1$, define the formulas $\varphi^1_{=j}$, $\varphi^i_{=j}$ via

$$\begin{aligned}
\varphi^1_{=0}(X_1,\dots,X_c\colon \bullet) &= \forall(Y_1,\dots,Y_c\colon \bullet).\, \varphi^1_<(X_1,\dots,X_c,Y_1,\dots,Y_c) \vee \varphi^1_=(X_1,\dots,X_c,Y_1,\dots,Y_c)\\
\varphi^1_{=j+1}(X_1,\dots,X_c\colon \bullet) &= \exists(Y_1,\dots,Y_c\colon \bullet).\, \varphi^1_{=j}(Y_1,\dots,Y_c) \wedge \varphi^1_{\text{succ}}(X_1,\dots,X_c,Y_1,\dots,Y_c)\\
\varphi^i_{=0}(X\colon \tau_i) &= \forall(Y\colon \tau_i).\, \varphi^i_<(X,Y) \vee \varphi^i_=(X,Y)\\
\varphi^i_{=j+1}(X\colon \tau_i) &= \exists(Y\colon \tau_i).\, \varphi^i_{=j}(X,Y) \wedge \varphi^i_{\text{succ}}(X,Y)
\end{aligned}$$

where $\varphi^0_{=j}$ and $\varphi^i_{=j}$ express that $(X_1,\dots,X_c)$, resp. X is the $j+1$st element of the total order induced by $\varphi^{0]}_<$, resp. $\varphi^i_<$, if such an element exists. Clearly, the size of these formulas is linear in j.

4.2 The Reduction

Let $k \geq 1$ and let $\mathscr{M} = (Q,\Sigma,\Gamma,\square,\delta,q_{\text{init}},q_{\text{acc}},q_{\text{rej}})$ be a $2^k_{p(n)}$-space-bounded DTM that decides a query $\mathscr{Q}_\mathscr{M}$ over ordered LTS, i.e., it accepts those $w = \langle T \rangle$ for which $T \in \mathscr{Q}$. W.l.o.g. $p(n) = c \cdot n^{c-1}$ for some c, whence also for $n \geq c$ we have $p(n) \leq n^c$. We also assume that $\mathscr{M}$ rejects all inputs that do not encode an LTS ordered by a relation $<$.

We have to build a HO^{k+1}+PFP formula $\varphi(X_1,\dots,X_d)$ such that $T \models \varphi$ iff $T \in \mathscr{Q}_\mathscr{M}$ and T is ordered by $<$.

Encoding Configurations. Let $\tau = \bullet, \tau_{k+1}, \tau_{k+1}, \bullet$. Let T be an LTS ordered by $<$ such that its state set satisfies $|S| \geq \max\{c, |Q|, |\Gamma|\}$. Hence, $|S|^c \geq p(|S|)$. W.l.o.g. Q and Γ are ordered, i.e., $Q = \{q_0,\dots,q_{|Q|-1}\}$ and $\Gamma = \{\gamma_0,\dots,\gamma_{|\Gamma|-1}\}$. Since S is ordered by $<$, for each $q_i \in Q$ and for each $\gamma_j \in \Gamma$, there are unique states s_q and s_γ, given as the $i+1$st, resp. $j+1$st states in the total order $<$. An element of $[\![\tau]\!]^T$ has the form (s,H,I,s') with $s,s' \in [\![\bullet]\!]^T = S$ and $H,I \in [\![\tau_{k+1}]\!]^T$.

Definition 5. Let $M \in [\![(\tau)]\!]^T$. We say that M encodes a configuration $C = (q,h,t)$ of $\mathscr{M}$ if the following are true:

1. For all $(s,H,I,s') \in M$, we have that $s = s_q$.
2. For all $(s,H,I,s'),(t,H',I',t') \in M$, we have $s = t$ and $H = H'$ and H is the $h+1$st element in the total order induced by $\varphi^{k+1}_<$.
3. For each $I \in [\![\tau_{k+1}]\!]^T$, there is exactly one tuple of the form (s,H,I,s') in M.
4. If $j \leq 2^{p(|S|)}_k$, if I is the $j+1$st element in the total order induced by $\varphi^{k+1}_<$, and if $(s,H,I,s') \in M$, then $s' = s_\gamma$ for some $\gamma \in \Gamma$ and $t(j) = \gamma$.

The intuition here is the following: Since all tuples in M agree on s_q and H, this uniquely determines q and h. Moreover, since for each $I \in [\![\tau_{k+1}]\!]^T$, there is exactly one tuple of the form (s,H,I,s') in M, this defines a function $[\![\tau_{k+1}]\!]^T \to \Gamma$, and since $[\![\tau_{k+1}]\!]^T$ is linearly ordered via $\varphi^{k+1}_{<}$ and has cardinality $2^{p(|S|)}_k$ due to Lem. 4, this yields a function $\{0,\dots,2^{|S|^c}_k - 1\} \to \Gamma$. Since $\mathscr{M}$ is $2^{p(n))}_k$-space-bounded, all configurations of a run of $\mathscr{M}$ on input $\langle T,(s_1,\dots,s_d)\rangle$ have a head position less than $2^{p(|S|)}_k \leq |[\![\tau_{k+1}]\!]^T|$ and, consequently, all tape cells of such a configuration with index at least $2^{p(|S|)}_k$ must contain $\square$. Hence, such a set in $[\![(\tau)]\!]^T$ can encode any configuration $\mathscr{M}$ may enter during its run on input $\langle T,(s_1,\dots,s_d)\rangle$.

Now let $w = \langle T\rangle$. Consider the HO^{k+1}+PFP formula

$$\varphi^w_{\text{init}}(Y_q\colon \bullet, H\colon \tau_{k+1}, I\colon \tau_{k+1}, Y_\gamma\colon \bullet) = \varphi^{k+1}_{=0}(H) \wedge Y_q = s_{q_{\text{init}}} \wedge \bigwedge_{i=0}^{|w|-1} \varphi^{k+1}_{=i}(I) \to Y_\gamma = s_{w_i}$$
$$\wedge\, \exists(Z\colon \tau_{k+1})\ \varphi^{k+1}_{=|w|-1}(Z) \wedge \varphi^{k+1}_{<}(Z,I) \to Y_\gamma = s_\square.$$

It is of polynomial size and expresses that the tuple encoded in the variables Y_q, H, I, Y_γ is in the unique set that encodes the initial configuration of $\mathscr{M}$ on input w. We use shorthand such as $Y_q = s_{q_{\text{init}}}$ to abbreviate $\varphi^1_{=j}(Y_q)$ if q_{init} is the $j+1$st state w.r.t. $<$ on S.

The Partial Fixpoint. Consider the formula

$$\begin{aligned}
\psi_{\text{trans}}(X,Y_q,H,I,Y_\gamma) = {} & \exists(Y'_q,H',I',Y'_\gamma\colon \bullet,\tau_{k+1},\tau_{k+1},\bullet).\\
& \exists(Y''_q,H'',I'',Y''_\gamma\colon \bullet,\tau_{k+1},\tau_{k+1},\bullet).\\
& X(Y'_q,H',I',Y'_\gamma) \wedge X(Y''_q,H'',I'',Y''_\gamma) \wedge \varphi^{k+1}_{=}(H',I'') \wedge \varphi^{k+1}_{=}(I,I')\\
& \wedge \neg\varphi^{k+1}_{=}(H,I) \to \varphi^1_{=}(Y_\gamma,Y'_\gamma)\\
& \wedge \bigwedge_{(q',\gamma,q'',\gamma',L)\in\delta} Y'_q = s_{q'} \wedge Y'_\gamma = s_\gamma \to Y = s_{q''} \wedge \varphi^{k+1}_{\text{succ}}(H,H') \wedge \varphi^{k+1}_{=}(H,I) \to Y_\gamma = s_{\gamma'}\\
& \wedge \bigwedge_{(q',\gamma,q'',\gamma',N)\in\delta} Y'_q = s_{q'} \wedge Y'_\gamma = s_\gamma \to Y = s_{q''} \wedge \varphi^{k+1}_{=}(H,H') \wedge \varphi^{k+1}_{=}(H,I) \to Y_\gamma = s_{\gamma'}\\
& \wedge \bigwedge_{(q',\gamma,q'',\gamma',R)\in\delta} Y'_q = s_{q'} \wedge Y'_\gamma = s_\gamma \to Y = s_{q''} \wedge \varphi^{k+1}_{\text{succ}}(H',H) \wedge \varphi^{k+1}_{=}(H,I) \to Y_\gamma = s_{\gamma'}
\end{aligned}$$

where by abuse of syntax we write $(q',\gamma,q'',\gamma',L) \in \delta$ instead of $\delta(q',\gamma) = (q'',\gamma',L)$ etc.

Lemma 6. *Assume that $M \in [\![(\tau)]\!]^T$ with $\tau = (\bullet,\tau_{k+1},\tau_{k+1},\bullet)$ as before encodes some configuration C of the computation of $\mathscr{M}$ on input w, and assume that M' of the same type encodes the successor configuration of C.*

Let $s,s' \in S$ and $M_H, M_I \in [\![\tau_{k+1}]\!]^T$. Then

$$T,\eta[X \mapsto M, Y_q \mapsto s, H \mapsto M_H, I \mapsto I_H, Y_\gamma \mapsto s'] \models \psi_{trans} \quad \textit{iff} \quad (s,M_H,M_I,s') \in M'.$$

The intuition here is that ψ_{trans} defines the encoding of a successor of some configuration C from the encoding of C itself. The first existential quantifier requires the existence of a tuple in X that encodes the value of the tape at the same position as the new tuple will, i.e., they both must have the same third component, and the second quantifier requires the existence of a tuple that encodes the content of the tape at the head position. The third line enforces these properties. The fourth line fixes tape contents

not under the head. The last three lines, separated for the ease of notation, enforce that both the state transition and the new content of the tape at the old head position obey the transition function.

We now have the required machinery to encode a computation of $\mathscr{M}$ on input w into HO^{k+1}+PFP. Let

$$\begin{aligned}\psi(X,Y_q,H,I,Y_\gamma) &= \varphi_{\text{trans}}(X,Y_q,H,I,Y_\gamma)\\ &\quad \vee \forall(Y_q',H',I',Y_\gamma'\colon \bullet,\tau_{k+1},\tau_{k+1},\bullet).\neg X(Y_q',H',I',Y_\gamma') \wedge \varphi_{init}^{w}(Y_q,H,I,Y_\gamma)\\ \varphi_{\mathscr{M}} &= \varphi^{<} \wedge \exists(Y_q',H',I',Y_\gamma'\colon \bullet,\tau_{k+1},\tau_{k+1},\bullet).\, Y_q' = s_{q_{\text{acc}}} \wedge (\text{PFP}(X\colon(\tau).\,\psi)(Y_q',H',I',Y_\gamma').\end{aligned}$$

where $\varphi^{<}$ expresses that $<$ is a total order.

Lemma 7. *Let $p(n) = cn^{c-1}$ and let $\mathscr{M}$ be a $2_k^{p(n)}$-space-bounded DTM that decides a query over ordered LTS. Let Q be its state set and let Γ be its tape alphabet. Let T be an LTS ordered by $<$ and such that its state set satisfies $|S| \geq \max\{|Q|,|\Gamma|,c\}$. Let $w = \langle T \rangle$. Then*

$$T \models \varphi_{\mathscr{M},w} \text{ iff } w \in L(\mathscr{M}).$$

This follows from the previous lemmas. ψ stipulates that either X is empty, and a tuple is in its "return value" iff it is in the encoding of the initial configuration, using $\varphi_{\text{init}}^{w}$, or it defers to φ_{trans}. The formula $\varphi_{\mathscr{M}}$ then encodes the unique run of $\mathscr{M}$ on input w, by asking whether a tuple containing the accepting state is contained in the partial fixpoint of ψ. This is the case if and only if the machine halts in the accepting state, due to Lem. 6 and our observations on $\varphi_{\text{init}}^{w}$.

We omit the tedious, but standard argument that $\varphi_{\text{init}}^{w}$ can be rewritten into some φ_{init} not depending on w that internalizes the translation from T to $\langle T \rangle$.

Theorem 8. *HO^{k+1}+PFP captures k-EXPSPACE over ordered LTS for $k \geq 2$.*

One direction is by Thm. 3, the other direction is by the previous Lem. 7 plus the observation that LTS that are smaller than in the requirements of the lemma can be enumerated in a constant-size formula.

5 Conclusion

We have shown that, over ordered structures, the queries expressible in HO^{k+1}+PFP are exactly those decided by a $2_k^{p(n)}$-space-bounded DTM, i.e., that HO^{k+1}+PFP captures k-EXPSPACE over ordered structures for $k \geq 0$, extending the same result by Vardi for $k = 0$ [10].

It should be noted that the requirement that the structures in question be ordered can be removed for $k \geq 1$, as HO^{2}+PFP and above possess sufficient expressive power to "guess" an order, cf. Fagin's Theorem [3].

Our result has applications in descriptive complexity. Otto's Theorem [8] characterizes *bisimulation-invariant* P-queries as exactly those expressible in the polyadic modal *mu*-calculus. Contrary to Immerman's and Vardi's characterization [5, 10] of PTIME, the crucial requirement that the LTS be ordered is absent from this result, since an order can be recuperated in the bisimulation-invariant setting. However, the result makes use of the Immerman-Vardi Theorem. We have extended this result to a characterization of bisimulation-invariant k-EXPTIME [2] using Freire and Martin's characterization of k-EXPTIME [4], i.e., their generalization of the Immerman-Vardi Theorem. The results of this paper open up a similar characterization of the bisimulation-invariant exponential-space hierarchy, following from the second author's Master's thesis [7].

References

[1] S. Abiteboul & V. Vianu (1995): *Computing with First-Order Logic*. *J. Comput. Syst. Sci.* 50(2), pp. 309–335, doi:10.1006/JCSS.1995.1025.

[2] F. Bruse, D. Kronenberger & M. Lange (2022): *Capturing Bisimulation-Invariant Exponential-Time Complexity Classes*. In P. Ganty & D. Della Monica, editors: *Proc. 13th Int. Symp. on Games, Automata, Logics and Formal Verification, GandALF 2022*, *EPTCS* 370, pp. 17–33, doi:10.4204/EPTCS.370.2.

[3] R. Fagin (1974): *Generalized First-Order Spectra and Polynomial-Time Recognizable Sets*. *Complexity and Computation* 7, pp. 43–73.

[4] C. M. Freire & A. T. Martins (2011): *The Descriptive Complexity of the Deterministic Exponential Time Hierarchy*. In: *Proc. 5th Workshop on Logical and Semantic Frameworks with Applications, LSFA'10*, 269, pp. 71–82, doi:10.1016/j.entcs.2011.03.006.

[5] N. Immerman (1986): *Relational Queries Computable in Polynomial Time*. *Information and Control* 68(1-3), pp. 86–104, doi:10.1016/S0019-9958(86)80029-8.

[6] N. Immerman (1987): *Languages That Capture Complexity Classes*. *SIAM Journal of Computing* 16(4), pp. 760–778, doi:10.1137/0216051.

[7] D. Kronenberger (2018): *Capturing Bisimulation-Invariant Complexity Classes by Polyadic Higher-Order Fixpoint Logic*. Master's thesis, University of Kassel.

[8] M. Otto (1999): *Bisimulation-invariant PTIME and higher-dimensional μ-calculus*. *Theor. Comput. Sci.* 224(1-2), pp. 237–265, doi:10.1016/S0304-3975(98)00314-4.

[9] L. J. Stockmeyer (1976): *The polynomial-time hierarchy*. *TCS* 3(1), pp. 1–22, doi:10.1016/0304-3975(76)90061-X.

[10] M. Y. Vardi (1982): *The Complexity of Relational Query Languages (Extended Abstract)*. In: *Proc. 14th Symp. on Theory of Computing, STOC'82*, ACM, San Francisco, CA, USA, pp. 137–146, doi:10.1145/800070.

The μ-calculus' Alternation Hierarchy is Strict over Non-Trivial Fusion Logics

Leonardo Pacheco*
TU Wien
Vienna, Austria
leonardo.pacheco@tuwien.ac.at

The modal μ-calculus is obtained by adding least and greatest fixed-point operators to modal logic. Its alternation hierarchy classifies the μ-formulas by their alternation depth: a measure of the codependence of their least and greatest fixed-point operators. The μ-calculus' alternation hierarchy is strict over the class of all Kripke frames: for all n, there is a μ-formula with alternation depth $n+1$ which is not equivalent to any formula with alternation depth n. This does not always happen if we restrict the semantics. For example, every μ-formula is equivalent to a formula without fixed-point operators over S5 frames. We show that the multimodal μ-calculus' alternation hierarchy is strict over non-trivial fusions of modal logics. We also comment on two examples of multimodal logics where the μ-calculus collapses to modal logic.

1 Introduction

The modal μ-calculus is obtained by adding least and greatest fixed-point operators to modal logic. One measure of complexity for μ-formulas is their alternation depth, which measures the codependence of least and greatest fixed-point operators. Bradfield [3] showed that the μ-calculus' alternation hierarchy is strict: for all $n \in \mathbb{N}$, there is a formula with alternation depth $n+1$ which is not equivalent over unimodal frames to any formula with alternation depth n. On the other hand, Alberucci and Facchini [2] proved that, over S5 frames, every μ-formula is equivalent to a formula without fixed-point operators. See Chapter 2 of [11] for a survey on the μ-calculus' alternation hierarchy over various classes of frames.

Let L_0 and L_1 be modal logics with disjoint signatures. The fusion $\mathsf{L}_0 \otimes \mathsf{L}_1$ is the smallest modal logic containing both L_0 and L_1. If L_0 and L_1 are respectively characterized by the Kripke frames in F_0 and F_1, then the fusion $\mathsf{L}_0 \otimes \mathsf{L}_1$ is characterized by frames which are in F_i when restricted to the signature of L_i, for $i = 0, 1$. Fusion logics are commonly used for multi-agent epistemic logics and on the specification of computer systems. We show that, over fusions of non-trivial classes of frames, the μ-calculus' alternation hierarchy is strict. Our proof is based on work of Bradfield [3] and Alberucci [1].

Let F be a class of unimodal Kripke frames. We say $\circ \leftarrow \circ \rightarrow \circ$ is a subframe of F iff there is some frame $F = \langle W, R \rangle \in \mathsf{F}$ with pairwise different $w_0, w_1, w_2 \in W$ such that $w_0 R w_1$ and $w_0 R w_2$. We analogously define $\circ \rightarrow \circ \rightarrow \circ$ is a subframe of F and $\circ \rightarrow \circ$ is a subframe of F. We will define multimodal versions W_n of the winning region formulas W'_n to prove:

Main Theorem. *Let F_0, F_1, and F_2 be classes of unimodal Kripke frames closed under isomorphic copies and disjoint unions. If*

1. *$\circ \leftarrow \circ \rightarrow \circ$ is a subframe of F_0 and $\circ \rightarrow \circ$ a subframe of F_1; or*

*I would like to thank Thibaut Kouptchinky for comments and proof-reading. I would also like to thank the reviewers for their comments. This research was partially funded by the FWF grant TAI-797.

Alexis Saurin (Editor): Fixed Points
in Computer Science 2024 (FICS 2024).
EPTCS 435, 2025, pp. 93–103, doi:10.4204/EPTCS.435.8

2. $\circ \to \circ \to \circ$ *is a subframe of* F_0 *and* $\circ \to \circ$ *a subframe of* F_1*;*

then the μ*-calculus' alternation hierarchy is strict over* $\mathsf{F}_0 \otimes \mathsf{F}_1$*. If*

3. $\circ \to \circ$ *is a subframe of* F_0*,* F_1*, and* F_2*;*

then the μ*-calculus' alternation hierarchy is strict over* $\mathsf{F}_0 \otimes \mathsf{F}_1 \otimes \mathsf{F}_2$*.*

Corollary. *Let* $\{\mathsf{L}_0, \mathsf{L}_1\} \subseteq \{\mathsf{K}, \mathsf{K4}, \mathsf{S4}, \mathsf{KD45}, \mathsf{S5}, \mathsf{GL}\}$*, then the* μ*-calculus' alternation hierarchy is strict over* $\mathsf{L}_0 \otimes \mathsf{L}_1$*.*

One proof of the strictness of the μ-calculus makes essential use of parity games [3, 1]. In this proof, evaluation games for the μ-calculus are encoded as parity games, parity games are encoded as Kripke models, and formulas defining winning regions for parity games are given as witnesses for the strictness. For the multimodal case, we need to make changes for both of these. The encoding of parity games gets more complicated as we cannot just use the graph of the game as the graph of the Kripke model, and need to use copies of frames from both classes along with auxiliary propositional symbols in the encoding. This also complicates the winning region formulas, which need to take into account these auxiliary propositional symbols.

While the hypotheses of the Main Theorem looks *ad hoc*, we conjecture that they are optimal.

Conjecture. *Let* F_0 *and* F_1 *be classes of unimodal Kripke frames closed under isomorphic copies and disjoint unions. Suppose* $\circ \to \circ$ *is a subframe of* F_0 *and* F_1*. Then every* μ*-formula is equivalent to one with alternation depth* 1 *over* $\mathsf{F}_0 \otimes \mathsf{F}_1$*.*

As a counterpoint, we comment on two multimodal logics where the μ-calculus collapses to modal logic. GLP is a provability logic which contains countably many modal operators; its fixed-point property was proved by Ignatiev [7]. $\mathsf{IS5}$ is an intuitionistic version of $\mathsf{S5}$ which can be thought of as a fragment of a bimodal logic; the μ-calculus' collapse to modal logic over $\mathsf{IS5}$ was proved by Pacheco [12].

Outline In Section 2, we review some basic definitions. In Sections 3, 4, and 5, we give a detailed proof of Item 1 of the Main Theorem: we first show that evaluation games for the μ-calculus are also parity games, then define the formulas W_n and show how parity games can be encoded as multimodal Kripke models and, at last, show that W_n is not equivalent to any formula with lower alternation depth. In Section 6, we sketch how to modify the proof to show Items 2 and 3 of the Main Theorem. In Section 7, we describe two examples of multimodal logics where the μ-calculus collapses to modal logic.

2 Preliminaries

The μ-calculus Fix a set Prop of propositional symbols, a set Var of variable symbols, and a non-empty signature Λ. The μ*-formulas* are generated by the following grammar:

$$\varphi := P \mid \neg P \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box_i \varphi_i \mid \Diamond_i \varphi \mid \mu X.\varphi \mid \nu X.\varphi,$$

where $P \in \text{Prop}$, $X \in \text{Var}$ is a variable symbol, and $i \in \Lambda$. We write $\eta X.\varphi$ for $\mu X.\varphi$ or $\nu X.\varphi$. The *set of subformulas* of a formula φ is denoted by $\text{Sub}(\varphi)$.

Given a signature Λ, a *Kripke frame* is a pair $M = \langle W, \{R_i\}_{i \in \Lambda} \rangle$ where: W is the set of *possible worlds*; and each R_i is a binary relation on W, the *accessibility relations*. A *Kripke model* is a triple $M = \langle W, \{R_i\}_{i \in \Lambda}, V \rangle$ obtained by extending a Kripke frame with a function V from propositional symbols to subsets of W; V is called a *valuation function*. Given a set $A \subseteq W$, the *augmented model* $M[X := A]$ is

obtained by setting $V(X) := A$. A *pointed Kripke model* is a pair (M,w) consisting of a Kripke model M and a world w of M.

Fix a Kripke model $M = \langle W, \{R_i\}_{i\in\Lambda}, V\rangle$. Given a μ-formula $\varphi(X)$ with a distinguished variable X, let $\Gamma_{\varphi(X)} : \mathscr{P}(W) \to \mathscr{P}(W)$ be the operator which maps $A \subseteq W$ to $\|\varphi(X)\|^{M[X:=A]}$. We define the valuation $\|\varphi\|^M$ on M inductively on the structure of μ-formulas:

- $\|P\|^M := V(P)$;
- $\|X\|^{M[X:=A]} := A$;
- $\|\varphi \wedge \psi\|^M := \|\varphi\|^M \cap \|\psi\|^M$;
- $\|\Box_i\varphi\|^M := \{w \in W \mid \forall v.wR_iv \to v \in \|\varphi\|^M\}$;
- $\|\mu X.\varphi\|^M$ is the least fixed-point of $\Gamma_{\varphi(X)}$;
- $\|\neg\varphi\|^M := W \setminus \|\varphi\|^M$;
- $\|\varphi \vee \psi\|^M := \|\varphi\|^M \cup \|\psi\|^M$;
- $\|\Diamond_i\varphi\|^M := \{w \in W \mid \exists v.wR_iv \wedge v \in \|\varphi\|^M\}$;
- $\|\nu X.\varphi\|^M$ is the greatest fixed-point of $\Gamma_{\varphi(X)}$.

Note that the operator $\Gamma_{\varphi(X)}$ is monotone for all formula $\varphi(X)$: if $A \subseteq B \subseteq W$, then $\Gamma_{\varphi(X)}(A) \subseteq \Gamma_{\varphi(X)}(B)$. By the Knaster–Tarski Theorem, the least and greatest fixed-points of $\Gamma_{\varphi(X)}$ are well-defined. We say a formula φ is *valid on a Kripke model M* iff φ holds on all worlds of M. We say a formula φ is *valid on a Kripke frame F* iff φ is valid on all Kripke models obtained by adding valuations to F. When convenient, we write $M,w \models \varphi$ for $w \in \|\varphi\|^M$. See [4] for more information on the μ-calculus.

Fusions Fix $n \in \mathbb{N}$. A (normal) modal logic is a set of formulas (without fixed-point operators) closed containing all the propositional tautologies and closed under *modus ponens*, necessitation, and substitution. Let $\{\mathsf{L}_j\}_{j\le n}$ be a collection of modal logics with pairwise disjoint signatures. The fusion $\bigotimes_{j\le n}\mathsf{L}_j$ is the smallest modal logic containing the logics $\{\mathsf{L}_j\}_{j\le n}$. Let $\{\mathsf{F}_j\}_{j\le n}$ be classes of frames with pairwise disjoint signatures $\{\Lambda_j\}_{j\le n}$. Put $\Lambda = \bigcup_{j\le n}\Lambda_j$. Define $\bigotimes_{j\le n}\mathsf{F}_j$ as the class of frames $F = \langle W, \{R_i\}_{i\in\Lambda}\rangle$ such that $\langle W, \{R_i\}_{i\in\Lambda_j}\rangle$ is a frame of F_j for all $j \le n$.

Suppose the modal logic L_j is characterized by the class of frames F_j, for all $j \le n$. Then $\bigotimes_{j\le n}\mathsf{L}_j$ is characterized by $\bigotimes_{j\le n}\mathsf{F}_j$. Furthermore, if all the L_j have the finite model property, then $\bigotimes_{j\le n}\mathsf{L}_j$ also has the finite model property. Similarly, if all the L_j are decidable, so is $\bigotimes_{j\le n}\mathsf{L}_j$. On the other hand, fusions do not preserve the complexity of the logics: almost all interesting fusions are PSPACE-hard. See [9, 5] for more on fusions of modal logics and other combinations of modal logics.

Alternation Hierarchy The μ-calculus' alternation hierarchy classifies the μ-formulas according to the co-dependence of its least and greatest fixed-point operators. We define it as follows:

- $\Sigma^\mu_0 (= \Pi^\mu_0)$ is the set of all μ-formulas with no fixed-point operators.
- Σ^μ_{n+1} is the closure of $\Sigma^\mu_n \cup \Pi^\mu_n$ under propositional operators, modal operators, μX, and the substitution: if $\varphi(X) \in \Sigma^\mu_{n+1}$ and $\psi \in \Sigma^\mu_{n+1}$ are such that no free variable of ψ becomes bound in $\varphi(\psi)$, then $\varphi(\psi) \in \Sigma^\mu_{n+1}$.
- Π^μ_{n+1} is the closure of $\Sigma^\mu_n \cup \Pi^\mu_n$ under propositional symbols, modal operators, νX, and the analogous substitution: if $\varphi(X) \in \Pi^\mu_{n+1}$ and $\psi \in \Pi^\mu_{n+1}$ are such that no free variable of ψ becomes bound in $\varphi(\psi)$, then $\varphi(\psi) \in \Pi^\mu_{n+1}$.

Let F be a class of Kripke frames. The μ-calculus' alternation hierarchy is strict over F iff, for all n, there is a formula in $\Sigma^\mu_{n+1} \cup \Pi^\mu_{n+1}$ which is not equivalent to any formula in $\Sigma^\mu_n \cup \Pi^\mu_n$ over F. The μ-calculus collapses to modal logic over F iff every μ-formula is equivalent to a formula without fixed-point operators over F.

Game Semantics The μ-calculus also has an equivalent game semantics. Fix a μ-formula φ, a Kripke model $M = \langle W, \{R_i\}_{i\in\Lambda}, V\rangle$, and a world w. For notational simplicity, we suppose each variable occurring in φ has only one occurrence and is bound by some fixed-point operator.[1] The evaluation game $\mathscr{G}(M, w \models \varphi)$ is a game for two players: Verifier and Refuter, denoted by V and R respectively. The positions of the game are of the form $\langle \psi, v\rangle$ with $\psi \in \mathrm{Sub}(\varphi)$ and $v \in W$. The initial position is $\langle \varphi, w\rangle$. Each position $\langle \psi, v\rangle$ is owned by a player, who makes the next move. Table 1 summarizes the ownership of $\langle \psi, v\rangle$ and admissible moves on it; both are determined by the construction of ψ. On Table 1, ψ_X denotes the unique subformula of φ such that X occurs freely in ψ_X and $\eta X.\psi_X \in \mathrm{Sub}(\varphi)$.

Let ρ be a run of an evaluation game $\mathscr{G}(M, w \models \varphi)$. If ρ is finite, V wins ρ iff R cannot make a move and R wins ρ iff V cannot make a move. If ρ is infinite, let $\eta X.\psi \in \mathrm{Sub}(\varphi)$ be a formula such that: positions of the form $\langle \eta X.\psi, v\rangle$ appear infinitely many often in ρ; and, for all formula θ such that positions $\langle \theta, v\rangle$ appear infinitely often in ρ, $\theta \in \mathrm{Sub}(\eta X.\psi)$. Then V wins ρ iff η is ν and R wins ρ iff η is μ. A strategy is a function indicating how a player should move. A winning strategy for V is a strategy σ for V such that V wins all runs where they follow σ. We define winning strategies for R similarly.

Relational semantics and game semantics are equivalent:

Proposition 1. *Let $M = \langle W, \{R_i\}_{i\in\Lambda}, V\rangle$ be a Kripke model, $w \in W$ be a world, and φ be a μ-formula. Then $M, w \models \varphi$ iff V has a winning strategy in the evaluation game $\mathscr{G}(M, w \models \varphi)$; and $M, w \not\models \varphi$ iff R has a winning strategy in the evaluation game $\mathscr{G}(M, w \models \varphi)$.*

Proof. See [4] or [12] □

Table 1: The rules of evaluation game for modal μ-calculus.

Verifier		Refuter	
Position	Admissible moves	Position	Admissible moves
$\langle \psi_1 \vee \psi_2, w\rangle$	$\{\langle \psi_1, w\rangle, \langle \psi_2, w\rangle\}$	$\langle \psi_1 \wedge \psi_2, w\rangle$	$\{\langle \psi_1, w\rangle, \langle \psi_2, w\rangle\}$
$\langle \Diamond_i \psi, w\rangle$	$\{\langle \psi, v\rangle \mid \langle w, v\rangle \in R_i\}$	$\langle \Box_i \psi, w\rangle$	$\{\langle \psi, v\rangle \mid \langle w, v\rangle \in R_i\}$
$\langle P, w\rangle$ and $w \notin V(P)$	$\emptyset$	$\langle P, w\rangle$ and $w \in V(P)$	$\emptyset$
$\langle \neg P, w\rangle$ and $w \in V(P)$	$\emptyset$	$\langle \neg P, w\rangle$ and $w \notin V(P)$	$\emptyset$
$\langle \mu X.\psi_X, w\rangle$	$\{\langle \psi_X, w\rangle\}$	$\langle \nu X.\psi_X, w\rangle$	$\{\langle \psi_X, w\rangle\}$
$\langle X, w\rangle$	$\{\langle \mu X.\psi_X, w\rangle\}$	$\langle X, w\rangle$	$\{\langle \nu X.\psi_X, w\rangle\}$

Parity games A parity game is a tuple $\mathscr{P} = \langle V_\exists, V_\forall, v_0, E, \Omega\rangle$ where two players $\exists$ and $\forall$ move a token in the graph $\langle V_\exists \cup V_\forall, E\rangle$. We suppose $V_\exists$ and $V_\forall$ are disjoint sets of *vertices*; $E \subseteq (V_\exists \cup V_\forall)^2$ is a set of *edges*; and $\Omega : V_\exists \cup V_\forall \to n$ is a *parity function*. If a player has no available move, then the other player wins. In an infinite play ρ, the winner is determined by the following parity condition: $\exists$ wins ρ iff the greatest parity which appears infinitely often in ρ is even; otherwise, $\forall$ wins ρ. $\exists$ wins the parity game $\mathscr{P}$ iff $\exists$ has a winning strategy; a winning strategy for $\exists$ is a function σ from $V_\exists$ to $V_\exists \cup V_\forall$, where, if $\exists$ follows σ, all resulting plays are winning for them. Similarly, $\forall$ wins $\mathscr{P}$ iff $\forall$ has a winning strategy.

[1] This statement is not problematic as we are interested in metamathematical properties of the μ-calculus. More care is needed when one is interested in the complexity of algorithms related to the μ-calculus. See [8].

Fix a parity game $\mathscr{P} = \langle V_\exists, V_\forall, v_0, E, \Omega\rangle$. The set of winning positions for $\exists$ in $\mathscr{P}$ is the set of positions v where $\exists$ wins the parity game if the players start at v. That is, $v \in V_\exists \cup V_\forall$ is a winning position for $\exists$ iff $\exists$ wins $\mathscr{P}_v = \langle V_\exists, V_\forall, v, E, \Omega\rangle$.

Sometimes it is convenient to suppose that all parity games are *tree-like*. That is, for all $v \in V_\exists \cup V_\forall$, there is no path $v = v_0 E \cdots E v_n = v$, for all $n \in \mathbb{N}$. Any parity game $\mathscr{P} = \langle V_\exists, V_\forall, v_0, E, \Omega\rangle$ can be unfolded into a tree-like parity game. In the unfolded game, instead of moving to a node v, the players move to a fresh copy of v. The unfolded parity game is bisimilar to the original game.

3 Evaluation games as parity games

Fix a model $M = \langle W, \{R_i\}_{i\in\Lambda}, V\rangle$, a world $w \in W$ and a μ-formula φ. We define a parity game $\mathscr{G}^{\mathrm{P}} = \mathscr{G}^{\mathrm{P}}(M, w \models \varphi) = \langle V_\exists, V_\forall, v_0, E, \Omega\rangle$ which is equivalent to $\mathscr{G} = \mathscr{G}(M, w \models \varphi)$.

The set of positions $V_\exists$ consists of the positions owned by V in $\mathscr{G}$. Similarly $V_\forall$ consists of the positions owned by $\forall$ in $\mathscr{G}$. The set of edges E consists of the transitions in $\mathscr{G}$. The initial position v_0 is $\langle \varphi, w\rangle$. Define the parity function:

- $\Omega(\langle \mu X.\psi, v\rangle) = 2(i+\varepsilon) - 1$ if $\mu X.\psi \in \Sigma^\mu_{2i+\varepsilon} \setminus \Pi^\mu_{2i+\varepsilon}$;
- $\Omega(\langle \nu X.\psi, v\rangle) = 2i$ if $\nu X.\psi \in \Pi^\mu_{2i+\varepsilon} \setminus \Sigma^\mu_{2i+\varepsilon}$;
- $\Omega(\langle \psi, v\rangle) = 0$ for ψ not of the form $\eta X.\psi$;

where $\varepsilon \in \{0,1\}$.

Proposition 2. *Let $M = (W, \{R_i\}_{i\in\Lambda}, V)$ be a Kripke model, $w \in W$, and φ a μ-formula. Then:*

$$\mathsf{V} \textit{ wins } \mathscr{G}(M, w \models \varphi) \iff \exists \textit{ wins } \mathscr{G}^{\mathrm{P}}(M, w \models \varphi).$$

Proof. Denote $\mathscr{G}(M, w \models \varphi)$ by $\mathscr{G}$ and $\mathscr{G}^{\mathrm{P}}(M, w \models \varphi)$ by $\mathscr{G}^{\mathrm{P}}$. As both games are on the same board, strategies for V and R in $\mathscr{G}$ are strategies for $\exists$ and $\forall$ in $\mathscr{G}^{\mathrm{P}}$. As any position is owned by V in $\mathscr{G}$ iff it is owned by $\exists$ in $\mathscr{G}$, any finite run is winning for V iff it is winning for $\exists$.

Consider an infinite run ρ. The parity $\Omega(\langle \psi, v\rangle)$ is odd iff $\psi \in \Sigma^k \setminus \Pi^k$ for some $k \in \mathbb{N}$. If the greatest infinitely often occurring parity in ρ is odd, then some $\mu X.\psi$ is the outermost infinitely often occurring fixed-point formula. Otherwise, if $\mu X.\psi \in \mathrm{Sub}(\nu \mathrm{Y}.\theta)$ and $\nu Y.\theta$ is the outermost infinitely occurring fixed-formula formula, then $\Omega(\langle \nu Y.\theta, v\rangle) \geq \Omega(\langle \mu X.\psi, v\rangle)$ and $\Omega(\langle \nu Y.\theta, v\rangle)$ is even. Similarly, if the greatest infinitely often occurring parity in ρ is even, then some $\nu X.\psi$ is the outermost infinitely often occurring fixed-point formula. Either way, ρ is winning for V in $\mathscr{G}$ iff ρ is winning for $\exists$ in $\mathscr{G}^{\mathrm{P}}$. □

4 Winning region formulas

Let F_0 and F_1 be classes of frames with signatures $\{0\}$ and $\{1\}$, respectively. Suppose that $\circ \leftarrow \circ \rightarrow \circ$ is a subframe of F_0 and that $\circ \rightarrow \circ$ is a subframe of F_1. Fix $F_0 \in \mathsf{F}_0$ and $F_1 \in \mathsf{F}_1$ witnessing these facts. Given a parity game $\mathscr{P}$ we will define an associated Kripke model $\mathscr{P}^{\mathrm{K}}$ with frame in $\mathsf{F}_0 \otimes \mathsf{F}_1$. We will also define winning region μ-formulas W_n, for all $n \in \mathbb{N}$. If $\mathscr{P}$ is a parity game which uses parities up to n, then $\exists$ wins $\mathscr{P}$ starting at v iff $\mathscr{P}^{\mathrm{K}}, v \models W_n$.

Let $\mathscr{P} = \langle V_\exists, V_\forall, v, E, \Omega\rangle$ be a parity game. We represent $\mathscr{P}$ as a birelational Kripke model $\mathscr{P}^K = \langle W, R_0, R_1, V\rangle$. The set W of possible worlds will consist of a world $\underline{v}$ for each state $v \in V_\exists, V_\forall$ and a countable supply of other worlds. If $v \in V_\exists, V_\forall$ and $vE = \{v_0, \ldots v_n\}$, then we will represent the connection between v and the v_i using fresh isomorphic copies of F_0 and F_1. We first use a copy of F_0 to choose

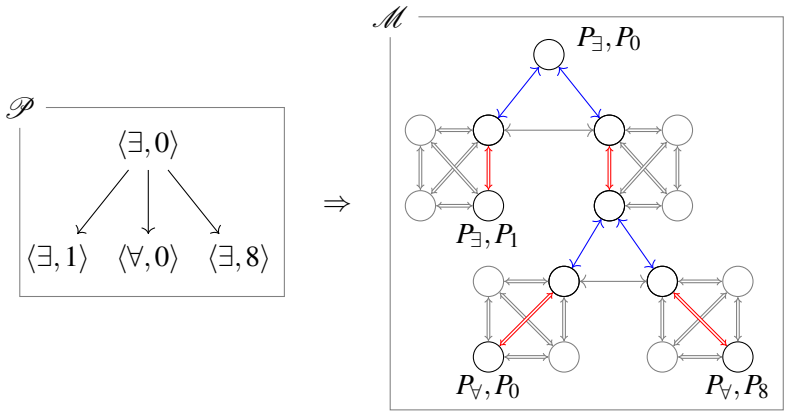


Figure 1: Example of a parity game $\mathscr{P}$ and the corresponding bimodal model $\mathscr{P}^K$. The model $\mathscr{P}^K$ is built using copies of S5 models.

between v_0 and the other vertices, then we use copies of F_1 to confirm the choices. Similarly, we use a copy of F_0 to choose between v_1 and the other vertices, and copies of F_1 to confirm the choices. We repeat this procedure until we use up all the v_i. By using fresh copies of F_0 and F_1, we guarantee that the resulting frame is in $\mathsf{F}_0 \otimes \mathsf{F}_1$. We denote by $\underline{v}, \underline{v_0}, \cdots$ the worlds of $\mathscr{P}^{\mathrm{K}}$ corresponding to the positions $v, v_0, \cdots$; we do not name the other worlds connecting them. An example of this construction is depicted in Figure 1.

We will use fresh propositional symbols bd, pos, pre_0, pre_1, nxt_0, and nxt_1 when defining $\mathscr{P}^K$. The proposition symbol bd indicate that a world is used to represent the parity game. That is, only the isomorphic copies of $\circ \leftarrow \circ \rightarrow \circ$ and $\circ \rightarrow \circ$ used in the paragraph above satisfy bd. The proposition symbol pos indicates that a world corresponds to a position in the parity game. That is, it holds only on worlds which are $\underline{v}$ for some $v \in V_\exists \cup V_\forall$. The proposition symbols pre_0, pre_1, nxt_0, and nxt_1 are used to represent the direction of the moves in the parity game in the Kripke model. pre_0 holds when we are making a choice and nxt_0 holds after we make a choice. Similarly, pre_1 holds when we are confirming a choice and nxt_1 holds after we confirmed a choice. These propositional symbols allow us to stay in the part of the model which represents the parity game. They will also guarantee that sequences moves in evaluation games over $\mathscr{P}^K$ correspond to moves in $\mathscr{P}$.

The proposition symbols $P_\exists$ and $P_\forall$ indicate the ownership of the positions: $P_\exists$ holds at $\underline{v}$ iff $v \in V_\exists$ and $P_\forall$ holds at $\underline{v}$ iff $v \in V_\forall$. The proposition symbols $P_0, \ldots, P_n$ indicate the parities of the positions: P_i holds at $\underline{v}$ iff $\Omega(v) = i$. At each $\underline{v}$, exactly one of the P_i will hold. The proposition symbols $P_\exists$, $P_\forall$, $P_0, \ldots, P_n$ are false at worlds which are not of the form $\underline{v}$ for some $v \in V_\exists \cup V_\forall$. This finishes the definition of $\mathscr{P}^{\mathrm{K}}$.

To define the winning region formulas W_n, we use the following shorthand formulas:

- $\blacklozenge\varphi := \nu Y.\mathrm{pre}_0 \wedge \mathrm{bd} \wedge \Diamond_0(\mathrm{nxt}_0 \wedge \mathrm{pre}_1 \wedge \mathrm{bd} \wedge \Diamond_1(\mathrm{nxt}_1 \wedge \mathrm{bd} \wedge ((Y \wedge \neg\mathrm{pos}) \vee (\varphi \wedge \mathrm{pos}))))$; and
- $\blacksquare\varphi := \nu Y.\mathrm{pre}_0 \wedge \mathrm{bd} \rightarrow \Box_0(\mathrm{nxt}_0 \wedge \mathrm{pre}_1 \wedge \mathrm{bd} \rightarrow \Box_1(\mathrm{nxt}_1 \wedge \mathrm{bd} \rightarrow ((Y \wedge \neg\mathrm{pos}) \wedge (\varphi \wedge \mathrm{pos}))))$,

where Y is a fresh variable symbol. We use these modalities to represent a move in $\mathscr{P}$ as multiple moves in evaluation game $\mathscr{P}^{\mathrm{K}}, \underline{v} \models W_n$. Given $n \in \mathbb{N}$, define:

$$W_n := \eta X_n \ldots \nu X_2 \mu X_1 \nu X_0. \bigvee_{0 \le j \le n} [(P_j \wedge P_\exists \wedge \blacklozenge X_j) \vee (P_j \wedge P_\forall \wedge \blacksquare X_j)].$$

The formula W_n defines the winning positions of $\exists$ in parity games using parities up to n:

Proposition 3. *Let* $\mathscr{P} = \langle V_\exists, V_\forall, v_0, E, \Omega \rangle$ *be a parity game. If* $\max\{\Omega(v) \mid v \in W\} \leq n$, *then*

$$\mathscr{P}^K, \underline{v}_0 \models W_n \text{ iff } \exists \text{ wins } \mathscr{P}.$$

Proof. Suppose $\mathscr{P}^{\mathrm{K}}, \underline{v}_0 \models W_n$. Let σ be a winning strategy for V in the evaluation game $\mathscr{G} := \mathscr{G}(\mathscr{P}^{\mathrm{K}}, \underline{v}_0 \models W_n)$. We define a winning strategy σ' for $\exists$ in $\mathscr{P}$ while playing simultaneous runs of $\mathscr{G}$ and $\mathscr{P}$.

The games $\mathscr{G}$ and $\mathscr{P}$ start at positions $\langle W_n, \underline{v}_0 \rangle$ and v_0, respectively. First, have the players move to the position

$$\left\langle \bigvee_{0 \leq j \leq n} [(P_j \wedge P_\exists \wedge \blacklozenge X_j) \vee (P_j \wedge P_\forall \wedge \blacksquare X_j), \underline{v}_0 \right\rangle$$

in $\mathscr{G}$.

Now, suppose the players are at positions

$$\left\langle \bigvee_{0 \leq j \leq n} [(P_j \wedge P_\exists \wedge \blacklozenge X_j) \vee (P_j \wedge P_\forall \wedge \blacksquare X_j), \underline{v} \right\rangle$$

in $\mathscr{G}$ and v in $\mathscr{P}$, respectively. As σ is winning for V in $\mathscr{G}$, σ does not make any immediately losing move. That is, V picks the disjuncts according to v's parity and owner. We also have $\forall$ make non-immediately losing moves. The players eventually reach one of two possible cases:

Case 1. The players are in the position $\langle \blacklozenge X_j, \underline{v} \rangle$ in $\mathscr{G}$, with $v \in V_\exists$. By our choice of σ, V must eventually reach a position of the form $\langle X_j, \underline{v'} \rangle$. Then σ' tells $\exists$ to move to v' in $\mathscr{P}$.

Case 2. The players are in the position $\langle \blacksquare X_j, \underline{v} \rangle$ in $\mathscr{G}$ and $v \in V_\forall$ in $\mathscr{P}$. If $\forall$ moves to v', R moves to $\langle X_j, \underline{v'} \rangle$ in $\mathscr{G}$ in finitely many steps.

Now, have the players regenerate X_j in $\mathscr{G}$ and move until they get to positions of the form

$$\left\langle \bigvee_{0 \leq j \leq n} [(P_j \wedge P_\exists \wedge \blacklozenge X_j) \vee (P_j \wedge P_\forall \wedge \blacksquare X_j), \underline{v'} \right\rangle \text{ and } v'$$

in $\mathscr{G}$ and $\mathscr{P}$, respectively. We are back to the initial situation, and we repeat this process to define σ'.

We consider parallel runs ρ in $\mathscr{G}$ and ρ' in $\mathscr{P}$ played according to σ and σ', respectively. Then either both runs are finite or both runs are infinite. If ρ' is finite, this means that one of the players didn't have a move available to play at a position v in $\mathscr{P}$. Therefore, one of the players couldn't find a valid position to play after $\langle \blacklozenge X_{\Omega(v)}, \underline{v} \rangle$ or $\langle \blacksquare X_{\Omega(v)}, \underline{v} \rangle$. The former is not possible by our choice of σ, so it must be $\forall$ who could not make a move. Therefore $\exists$ wins ρ'. If ρ is infinite, then the outermost infinitely often regenerated fixed-point operator is some νX_{2k}. By the construction of σ' the greatest infinitely often occurring parity must be $2k$. Therefore $\exists$ wins ρ'. We can now conclude that σ' is a winning strategy for $\exists$ in $\mathscr{P}$.

On the other hand, suppose $\exists$ wins $\mathscr{P}$ via σ'. We define σ for V in $\mathscr{G}$. At vertices of the form $\langle \blacklozenge X_j, \underline{v} \rangle$ in $\mathscr{G}$, have V move to

$$\sigma(\langle \blacklozenge X_j, \underline{v} \rangle) := \langle X_j, \underline{v'} \rangle,$$

with $v' = \sigma'(v)$. On other positions, have σ be the non-immediately losing moves for V.

Consider parallel runs ρ in $\mathscr{G}$ and ρ' in $\mathscr{P}$ played according to σ and σ', respectively. If ρ is finite, then one of the players made a move which invalidates one of the auxiliary propositions, or did not have an adequate moves after a position of the form $\langle \blacklozenge X_j, \underline{v} \rangle$ or $\langle \blacksquare X_j, \underline{v} \rangle$. By the choice of σ' and definition of σ, V makes no such move. So it must be R who made such move and lost the game; therefore V wins. If

ρ is infinite, the greatest parity appearing infinitely often in ρ' is even. Therefore the outermost infinitely often occurring fixed-point operator in ρ is a ν-operator. ρ is winning for V. Therefore σ is a winning strategy for V in $\mathscr{G}$. □

Given an evaluation game $\mathscr{G}(M,w \models \varphi)$, we define the Kripke model $\mathscr{G}^{\mathrm{K}}(M,w \models \varphi)$ as $(\mathscr{G}^{\mathrm{P}}(M,w \models \varphi))^{\mathrm{K}}$. As evaluation games are also parity games, the W_n also define winning regions for V in evaluation games:

Proposition 4. *Let $M = (W,R_0,R_1,V)$ be a bimodal Kripke model, $w \in W$, and φ a bimodal μ-formula. If $n \geq 1$ and the greatest parity used in $\mathscr{G}^{\mathrm{P}}(M,w \models \varphi)$ is less or equal than n, then:*

$$M,w \models \varphi \text{ iff } \mathscr{G}^{K}(M,w \models \varphi), \langle \varphi, w \rangle \models W_n.$$

Proof. We have:

$$\begin{aligned} M,w \models \varphi &\text{ iff } \mathsf{V} \text{ wins } \mathscr{G}(M,w \models \varphi) \\ &\text{iff } \exists \text{ wins } \mathscr{G}^{\mathrm{P}}(M,w \models \varphi) \\ &\text{iff } \mathscr{G}^{\mathrm{K}}(M,w \models \varphi), \langle \varphi, w \rangle \models W_n. \end{aligned}$$

The first equivalence follows from Proposition 1, the second one follows from Proposition 2, the third one follows from Proposition 3. □

5 Strictness

Fix classes of frames F_0 and F_1 with signatures $\{0\}$ and $\{1\}$, respectively. We show that, if $\circ \leftarrow \circ \rightarrow \circ$ is a subframe of F_0 and that $\circ \rightarrow \circ$ is a subframe of F_1, then the μ-calculus' alternation hierarchy is strict over $\mathsf{F}_0 \otimes \mathsf{F}_1$.

Let $(M,w) = \langle W,R_0,R_1,V,w \rangle$ and $(M',w') = \langle W',R'_0,R'_1,V',w' \rangle$ be pointed Kripke models without loops in their graphs. (M,w) is *isomorphic* to (M',w') iff there is a bijection $I : W \rightarrow W'$ such that:

- $I(w) = w'$;
- for all $v,v' \in W$, vR_0v' iff $I(v)R'_0I(v')$;
- for all $v,v' \in W$, vR_1v' iff $I(v)R'_1I(v')$; and
- for all $v \in W$, $v \in V(P)$ iff $I(v) \in V'(P)$.

For all $n \in \mathbb{N}$, let $(M \upharpoonright n,w)$ be the submodel of (M,w) obtained by restricting W to worlds with distance less than n from w. We say (M,w) is *n-isomorphic* to (M',w') if and only if $(M \upharpoonright n,w)$ is isomorphic to $(M' \upharpoonright n,w')$. For any (M,w), $(M \upharpoonright 0,w)$ is an empty Kripke model. We assume the empty Kripke model is isomorphic to itself.

Given a μ-formula φ, let f_φ be the function mapping a pointed model to the pointed Kripke model representing its evaluation game with respect to φ. That is $f_\varphi(M,w) = (\mathscr{G}^{\mathrm{K}}(M,w \models \varphi), \langle \varphi, w \rangle)$, for all pointed models (M,w).

Lemma 5. *Fix a μ-formula φ. If (M,w) and (M',w') are n-isomorphic via a function I, then $f_{\varphi \wedge \varphi}(M,w)$ and $f_{\varphi \wedge \varphi}(M',w')$ are $(n+1)$-isomorphic via the function J defined by:*

$$J(\langle \psi, w \rangle) = (\langle \psi, I(w) \rangle),$$

for all world w of M and subformula ψ of φ.

Proof. As (M,w) and (N,v) are n-isomorphic, the evaluation games $\mathscr{G}(M,w \models \varphi \wedge \varphi)$ and $\mathscr{G}(N,v \models \varphi \wedge \varphi)$ are going to be same up to n-many plays of the form $\langle \triangle \psi, w \rangle$, with $\triangle \in \{\Box_0, \Diamond_0, \Box_1, \Diamond_1\}$. As the first move in an evaluation game for the formula $\varphi \wedge \varphi$ is to choose between a conjunction, we can guarantee that the two games above are the same up to $n+1$ moves. □

Lemma 6. *For all μ-formula φ, the function $f_{\varphi\wedge\varphi}$ has a fixed-point (up to isomorphism). That is, there is a model (M,w) such that $f_\varphi(M,w)$ is isomorphic to (M,w).*

Proof. Let (M_0,w_0) be a fixed arbitrary pointed Kripke model. We define $(M_{n+1},w_{n+1}) = f_{\varphi\wedge\varphi}(M_n,w_n)$ inductively on $n \in \mathbb{N}$. If $n=0$, then (M_0,w_0) and (M_1,w_1) are trivially 0-isomorphic. By induction on n, (M_n,w_n) and (M_{n+1},w_{n+1}) are n-isomorphic via Lemma 5. Therefore, if $m > n$ then (M_n,w_n) is n-isomorphic to (M_m,w_m).

We can now define a pointed Kripke model (M,w) which is n-isomorphic to (M_n,w_n) for all n. We identify $(M_n \restriction n, w_n)$ and $(M_{n+1} \restriction n, w_{n+1})$, since they are n-isomorphic by the restriction of the isomorphism J_n from Lemma 5. Furthermore, the isomorphisms J_n and J_{n+1} coincide on $(M_n \restriction n, w_n)$ by construction. Let M be the Kripke model whose graph is the union of the graphs of the models $M_n \restriction n$ and whose valuation is the union of the valuation of the $M_n \restriction n$; also set $w = w_0$. Then $f_{\varphi\wedge\varphi}(M,w)$ is the Kripke model whose graph is the union of the graphs of the models $M_{n+1} \restriction n+1$ and whose valuation the union of the valuation of the $M_{n+1} \restriction n+1$. The union of the J_n is an isomorphism between (M,w) and $f_{\varphi\wedge\varphi}(M,w)$. □

Proof of Item 1 of the Main Theorem. Let F_0 and F_1 be classes of unimodal Kripke frames closed under isomorphic copies and disjoint unions. Suppose $\circ \leftarrow \circ \rightarrow \circ$ is a subframe of F_0 and $\circ \rightarrow \circ$ a subframe of F_1.

If n is even, then $W_n \in \Pi^\mu_{n+1}$. For a contradiction, suppose that W_n is equivalent to some formula in Π^μ_n over $\mathsf{F}_0 \otimes \mathsf{F}_1$. Let $\varphi \in \Sigma^\mu_n$ be equivalent to $\neg W_n$. Let (M,w) be a fixed-point of $f_{\varphi\wedge\varphi}$. Then

$$\begin{aligned} M,w \models \neg W_n &\iff M,w \models \varphi \wedge \varphi \\ &\iff f_{\varphi\wedge\varphi}(M,w) \models W_n \\ &\iff M,w \models W_n. \end{aligned}$$

The second equivalence follows from Proposition 3 and the third one follows from Lemma 6. This is a contradiction, and so W_n is not equivalent to any formula in Π^μ_n over $\mathsf{F}_0 \otimes \mathsf{F}_1$. The case for n odd is symmetric: $W_n \in \Sigma^\mu_{n+1}$ and is not equivalent to any formula in Σ^μ_n. □

6 Finishing the proof of the Main Theorem

To prove Items 2 and 3 of the Main Theorem, we modify two points in the proof above: first, we define new functions transforming parity games into Kripke models; second, we supply new versions of the modalities $\blacklozenge$ and $\blacksquare$.

We first consider the case of Item 2. Let F_0 and F_1 be classes of unimodal Kripke frames closed under isomorphic copies and disjoint unions. Suppose $\circ \rightarrow \circ \rightarrow \circ$ is a subframe of F_0 and $\circ \rightarrow \circ$ is a subframe of F_1. When we define a Kripke model $\mathscr{P}^{\mathrm{K}}$ from a parity game $\mathscr{P}$, we change the way we use the copies of $\circ \rightarrow \circ \rightarrow \circ$ and $\circ \rightarrow \circ$. Suppose $v \in V_\exists, V_\forall$ and $vE = \{v_0, \ldots v_n\}$. The players choose the next position as follows: they first move once in a copy of $\circ \rightarrow \circ \rightarrow \circ$; they then confirm some v_i using a copy of $\circ \rightarrow \circ$ or move along the current copy $\circ \rightarrow \circ \rightarrow \circ$; if they moved along $\circ \rightarrow \circ \rightarrow \circ$, they must confirm this move via a copy of $\circ \rightarrow \circ$.

To control the movement of the players over copies of $\circ \to \circ \to \circ$, we use three propositional symbols pre_0, mid_0, and nxt_0. Here, pre_0 holds at the first world of the copies of $\circ \to \circ \to \circ$, mid_0 holds at the second world, and nxt_0 holds at the third world. We define $\blacklozenge$ and $\blacksquare$ as follows:

- $\blacklozenge\varphi := \mu Y.\text{pre}_0 \wedge \text{bd} \wedge \Diamond_0[\text{mid}_0 \wedge \text{pre}_1 \wedge \wedge \text{bd} \wedge \Diamond_1(\text{nxt}_1 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))) \vee \Diamond_0(\text{nxt}_0 \wedge \text{pre}_1 \wedge \text{bd} \wedge \Diamond_1(\text{nxt}_1 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))))]$; and
- $\blacksquare\varphi := \mu Y.\text{pre}_0 \wedge \text{bd} \to \Box_0[\text{mid}_0 \wedge \text{pre}_1 \wedge \wedge \text{bd} \to \Box_1(\text{nxt}_1 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))) \wedge \Box_0(\text{nxt}_0 \wedge \text{pre}_1 \wedge \text{bd} \to \Box_1(\text{nxt}_1 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))))]$,

where Y is a fresh variable symbol. The definition of the winning region formulas W_n are the same as above, where $\blacklozenge$ and $\blacksquare$ use their new definitions.

Now for the proof of Item 3. Let F_0, F_1, and F_2 be classes of unimodal Kripke frames closed under isomorphic copies and disjoint unions. Suppose $\circ \to \circ$ is a subframe of F_0, F_1, and F_2. Given $v \in V_\exists, V_\forall$ and $vE = \{v_0, \ldots v_n\}$, we build a Kripke model as in the proof of Item 2, but instead of using a copy of $\circ \to \circ \to \circ$, we use two copies of $\circ \to \circ$, one from F_0 and one from F_1; we use copies of $\circ \to \circ$ from F_2 to confirm the choices. This time we will also use fresh proposition variables pre_2 and nxt_2 to control the movement of the players along copies of $\circ \to \circ$ in F_2. Here, we define $\blacklozenge$ and $\blacksquare$ as follows:

- $\blacklozenge\varphi := \mu Y.\text{pre}_0 \wedge \text{bd} \wedge \Diamond_0[\text{nxt}_0 \wedge \text{pre}_1 \wedge \text{pre}_2 \wedge \text{bd} \wedge \Diamond_2(\text{nxt}_2 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))) \vee \Diamond_1(\text{nxt}_1 \wedge \text{pre}_2 \wedge \text{bd} \wedge \Diamond_2(\text{nxt}_2 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))))]$; and
- $\blacksquare\varphi := \mu Y.\text{pre}_0 \wedge \text{bd} \to \Box_0[\text{nxt}_0 \wedge \text{pre}_1 \wedge \text{pre}_2\text{bd} \to \Box_2(\text{nxt}_2 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))) \wedge \Box_1(\text{nxt}_1 \wedge \text{pre}_2 \wedge \text{bd} \to \Box_2(\text{nxt}_2 \wedge \text{bd} \wedge ((Y \wedge \neg\text{pos}) \vee (\varphi \wedge \text{pos}))))]$,

where Y is a fresh variable symbol. The definition of the winning region formulas W_n are the same as above, where $\blacklozenge$ and $\blacksquare$ use their new definitions.

7 Case studies on the collapse over multimodal logics

We now comment on two logics where the μ-calculus collapses to modal logic. These are not originally framed in the context of multimodal μ-calculus.

Provability Logic GLP is a multimodal provability logic with signature $\mathbb{N}$, first defined by Japaridze. One of the possible arithmetical interpretations for each $\Box_n$ is as a provability predicate for $\mathsf{I}\Sigma_n$. Each modality $\Box_n$ satisfies the necessitation rule and the axioms for the provability GL: $\Box(P \to Q) \to (\Box P \to \Box Q)$ and $\Box(\Box P \to P) \to \Box P$. While GLP contains the fusion of infinitely many copies of GL, it is not a fusion logic: it also includes the axioms $\Box_m P \to \Box_n \Box_m P$, $\Diamond_m P \to \Box_n \Diamond_m P$, and $\Box_m P \to \Box_n P$, for all $m \leq n$.

Ignatiev [7] proved that GLP has the fixed-point property: if X is in the scope of some $\Box_i$ in $\varphi(X)$, then there is ψ such that $\mathsf{GLP} \vdash \psi \leftrightarrow \varphi(\psi)$. This implies that we do not get a more expressive logic if we add to it the operators μ and ν. While the additional conditions on the relation between the modalities makes it possible to have the fixed-point property, GLP is not complete over any class of Kripke models.

Intuitionistic Modal Logic IS5 is an intuitionistic variation of S5; it is also known as MIPQ. It consists of closure under necessitation and *modus ponens* of the set of formulas containing the intuitionistic tautologies along with the axioms $T := \Box\varphi \to \varphi \wedge \varphi \to \Diamond\varphi$, $4 := \Box\varphi \to \Box\Box\varphi \wedge \Diamond\Diamond\varphi \to \Diamond\varphi$, and $5 := \Diamond\varphi \to \Box\Diamond\varphi \wedge \Diamond\Box\varphi \to \Box\varphi$. An IS5 model is a tuple $\langle W, \preceq, R, V\rangle$ satisfying: $\preceq$ is a pre-order; R is an equivalence relation; $wR;\preceq v$ implies $w \preceq; Rv$; and $w \preceq v$ and $w \in V(P)$ implies $v \in V(P)$. IS5 can be

thought as a bimodal logic, where $\Box$ and $\Diamond$ are abbreviations for $\Box_{\preceq}\Box_R$ and $\Box_{\preceq}\Diamond_R$, respectively. Ono [10] and Fischer Servi [6] proved that IS5 is complete over IS5 frames.

Pacheco [12] proved that the μ-calculus collapses to constructive modal logic over IS5 frames using game semantics for the constructive μ-calculus. This example shows that, if we add restrictions on how we use multiple modalities, then we may still have the collapse to modal logic. Note that the relation $\langle W, \preceq \rangle$ is an S4 frame, and the μ-calculus does not collapse to modal logic over S4 frames [2]. So the restriction on the usage of the modalities here is quite strong.

Motivated by the examples above, we close the paper with a problem:

Problem. *When does the μ-calculus collapse to modal logic over multimodal frames?*

References

[1] Luca Alberucci (2002): *Strictness of the Modal μ-Calculus Hierarchy*. In Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors: *Automata Logics, and Infinite Games: A Guide to Current Research*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 185–201, doi:10.1007/3-540-36387-4_11.

[2] Luca Alberucci & Alessandro Facchini (2009): *The Modal μ-Calculus Hierarchy over Restricted Classes of Transition Systems*. *The Journal of Symbolic Logic* 74(4), pp. 1367–1400, doi:10.2178/jsl/1254748696.

[3] Julian C. Bradfield (1998): *Simplifying the Modal Mu-Calculus Alternation Hierarchy*. In G. Goos, J. Hartmanis, J. van Leeuwen, Michel Morvan, Christoph Meinel & Daniel Krob, editors: *STACS 98*, 1373, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 39–49, doi:10.1007/BFb0028547.

[4] Julian C. Bradfield & Igor Walukiewicz (2018): *The Mu-Calculus and Model Checking*. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith & Roderick Bloem, editors: *Handbook of Model Checking*, Springer International Publishing, Cham, pp. 871–919, doi:10.1007/978-3-319-10575-8_26.

[5] Walter Carnielli & Marcelo Esteban Coniglio (2020): *Combining Logics*. In Edward N. Zalta, editor: *The Stanford Encyclopedia of Philosophy*, fall 2020 edition, Metaphysics Research Lab, Stanford University.

[6] Gisèle Fischer Servi (1978): *The Finite Model Property for MIPQ and Some Consequences*. *Notre Dame Journal of Formal Logic* XIX(4), pp. 687–692.

[7] Konstantin N. Ignatiev (1993): *On Strong Provability Predicates and the Associated Modal Logics*. *Journal of Symbolic Logic* 58(1), pp. 249–290, doi:10.2307/2275337.

[8] Clemens Kupke, Johannes Marti & Yde Venema (2021): *On the Size of Disjunctive Formulas in the μ-Calculus*. *Electronic Proceedings in Theoretical Computer Science* 346, pp. 291–307, doi:10.4204/EPTCS.346.19.

[9] Agi Kurucz (2007): *Combining Modal Logics*. In Patrick Blackburn, Johan Van Benthem & Frank Wolter, editors: *Studies in Logic and Practical Reasoning*, *Handbook of Modal Logic* 3, Elsevier, pp. 869–924, doi:10.1016/S1570-2464(07)80018-8.

[10] Hiroakira Ono (1977): *On Some Intuitionistic Modal Logics*. *Publications of the Research Institute for Mathematical Sciences* 13(3), pp. 687–722, doi:10.2977/prims/1195189604.

[11] Leonardo Pacheco (2023): *Exploring the Difference Hierarchies on μ-Calculus and Arithmetic—from the Point of View of Gale–Stewart Games*. Ph.D. thesis, Tohoku University.

[12] Leonardo Pacheco (2023): *Game Semantics for the Constructive μ-Calculus*, doi:10.48550/arXiv.2308.16697. arXiv:2308.16697.