LICS 2012 Short Papers
# Table of Contents

## Short Papers Selected from Regular Submissions

A New Order-theoretic Characterisation of the Polytime Computable Functions ............................................. 7  
*Martin Avanzini, Naohi Eguchi and Georg Moser*

Views: Compositional Reasoning for Concurrent Programs ................................................................. 9  
*Thomas Dinsdale-Young, Lars Birkedal, Philippa Gardner, Matthew Parkinson and Hongseok Yang*

Herbrand-Confluence for Cut Elimination in Classical First Order Logic ......................................................... 11  
*Stefan Hetzl and Lutz Straßburger*

The Refined Calculus of Inductive Construction: Parametricity and Abstraction ............................................. 13  
*Chantal Keller and Marc Lasson*

Undecidable First-Order Theories of Affine Geometries .................................................................................. 15  
*Antti Kuusisto, Jeremy Meyers and Jonni Virtema*

Canonical Progress Measures for Parity Games ............................................................................................ 17  
*Konstantinos Mamouras*

Down the Borel Hierarchy: Solving Muller Games via Safety Games ......................................................... 19  
*Daniel Neider, Roman Rabinovich and Martin Zimmermann*

Interaction Graphs: Additives ......................................................................................................................... 21  
*Thomas Seiller*

Ribbon Proofs for Separation Logic ................................................................................................................ 23  
*John Wickerson, Mike Dodds and Matthew Parkinson*

## Short Papers from the Specific Short Papers Call

Logical Bell Inequalities ................................................................................................................................. 27  
*Samson Abramsky and Lucien Hardy*

Approximations of Conjunctive Queries ........................................................................................................ 29  
*Pablo Barceló, Leonid Libkin and Miguel Angel Romero Orth*

Complete Axiomatizations of Fragments of Monadic Second-Order Logic on Finite Trees .......................... 31  
*Amelie Gheerbrant and Balder ten Cate*

Uniform Polytime Computable Operators on Univariate Real Analytic Functions ...................................... 33  
*Akitoshi Kawamura, Norbert Müller, Martin Ziegler and Carsten Rösnick*

A metric analogue of Stone duality for Markov processes ........................................................................... 35  
*Kim Larsen, Radu Mardare and Prakash Panangaden*

The Comprehension Cube ............................................................................................................................... 37  
*Daniel Leivant*

On the expressivity of linear logic subsystems characterizing polynomial time ............................................ 39  
*Matthieu Perrinel*
Short Papers Selected from Regular Submissions
A New Order-theoretic Characterisation of the Polytime Computable Functions

Martin Avanzini*, Naohi Eguchi† and Georg Moser*

*Institute of Computer Science
University of Innsbruck, Austria

Email: {martin.avanzini, georg.moser}@uibk.ac.at

†Mathematical Institute
Tohoku University, Japan

Email: eguchi@math.tohoku.ac.jp

I. INTRODUCTION

In this work we are concerned with the complexity analysis of term rewrite systems (TRSs for short) and the ramifications of such an analysis in implicit computational complexity. Details and full proofs can be found in our technical report [1]. The foundation of rewriting is equational logic and term rewrite systems are conceivable as sets of directed equations. A natural way to measure the complexity of a TRS $\mathcal{R}$ is to measure the length of computations in $\mathcal{R}$. More precisely the runtime complexity of a TRS relates the maximal lengths of derivations to the size of the initial term, whose arguments are supposed to be values, i.e., irreducible. Indeed, runtime complexity is an invariant cost model [2]: all functions computed by a TRS allow realisations within polynomial overhead on a standard models of computation, like Turing machines.

We propose a new order, the small polynomial path order (sPOP for short) that induces polynomial (innermost) runtime complexities. The proposed order embodies the principle of predicative recursion set forth by Bellantoni and Cook [3] in their alternative characterisation $\mathcal{B}$ of the polynomial time computable functions (FP for short). Let us make this idea precise. We are assuming that the arguments of every function are partitioned into normal and safe ones. Notationally we write $f(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ where normal arguments are to the left, and safe arguments to the right of the semicolon. We define a restriction $\mathcal{B}_{\text{wsc}}$ of $\mathcal{B}$, where crucially the underlying composition scheme is replaced by a weaker form, disallowing composition under normal arguments. Defining rules of $\mathcal{B}_{\text{wsc}}$ are given in Fig. 1. The distinctive feature that permits only the formulation of feasible functions is of course the separation of normal (the recursion parameters) from safe arguments (the places for substituting recursive results). Due to a variation of a result by Handley and Wainer, we have that $\mathcal{B}_{\text{wsc}}$ contains all the polytime functions [4].

**Theorem 1.** Every function from $\mathcal{B}_{\text{wsc}}$ is polytime computable. Vice versa, $\mathcal{B}_{\text{wsc}}$ contains every polytime computable function.

Suppose the definition of a TRS $\mathcal{R}$ is based on the equations in $\mathcal{B}_{\text{wsc}}$. By its precise definition we can measure the runtime complexity based on the number $d$ of nested applications of safe recursion (SRN), that is, for such predicative recursive TRSs $\mathcal{R}$ of degree $d$, the runtime complexity is bounded by a polynomial of degree $d$. To decide whether a TRS $\mathcal{R}$ is predicative recursive of degree $d$, we have devised the order sPOP*. In fact this order theoretic characterisation is more liberal than $\mathcal{B}_{\text{wsc}}$, but main principles that allow a precise estimation of runtime complexities remain reflected.

II. MOTIVATION AND RELATED WORK

It is clear that an order-theoretic characterisation of predicative recursion is obtained as a restriction of the recursive path order. Predicative recursion stems from a careful analysis of primitive recursion and the recursive path order (with multiset status) characterises the class of primitive recursive functions [5]. The light multiset path order (LMPO for short) proposed by Marion [6] tames this recursive path order by embodying the principle of predicative recursion. LMPO captures FP, but captured rewrite systems do not admit polynomially runtime complexities in general. Polynomial runtime complexity analysis is an active research area in rewriting, see [7] for an overview. In particular, variations of dependency pairs for complexity analysis [8], [9], [10], also in combination with modularity results [11], established major breakthroughs. Since LMPO is inapplicable in this setting, the authors were motivated to present a restriction of LMPO, the polynomial path order (POP for short) [12]. This order is still complete for FP, but also induces feasible bounds on runtime complexities of rewrite systems. Still, POP* suffers from the fact that the precise degree of the polynomial certificate cannot be estimated. The order sPOP* eliminates this final weakness.

We also want to mention Bonfante et. al. [13] where restricted classes of polynomial interpretations are studied that can also precisely bind the runtime complexity of TRSs. Although incomparable to our technique, unarguably such semantic techniques admit a better intentionality, but are difficult to implement efficiently in an automated setting. In our complexity tool $TCT^1$ we see sPOP* as a fruitful and fast

---

1$TCT$ is open source. The interested reader might play with the web interface available at http://cl-informatik.uibk.ac.at/software/tct.
extension that handles systems in a fraction of a second.

III. THE SMALL POLYNOMIAL PATH ORDER

We consider constructor TRSs, where left-hand sides are of the form \( f(p_1, \ldots, p_n) \) with \( f \) so called defined symbols, and the patterns \( p_i \) are formed from variables and dedicated constructors. The latter symbols are used to form values. To precisely assess the complexity of a TRS, sPOP* allows recursive definitions only on a subset of defined symbols, the so called recursive symbols. Like recursive path orders, an instance of sPOP* is induced by a partial order on symbols, \( \succ \), the precedence, where \( f \succ g \) indicates that \( f \) is defined in terms of \( g \). Let \( f(s_1, \ldots, s_k ; \overline{s}) \succeq t \) hold if \( t \) is a subterm of a normal argument \( s_1, \ldots, s_k \). The next definition introduces small polynomial path orders \( \succ_{\text{pop}} \).

**Definition 2.** Let \( s = f(s_1, \ldots, s_k ; s_{k+1}, \ldots, s_{k+l}) \) and \( t \) be terms. Then \( s \succ_{\text{pop}^+} t \) if any of the following conditions hold:

1. \( s \succeq_{\text{pop}^+} t \) for some argument \( s_i \) of \( s \).
2. \( f \) is a defined symbol, \( t = g(t_1, \ldots, t_m ; t_{m+1}, \ldots, t_{m+n}) \).
3. \( f \succ g \) and the following conditions hold:
   - \( s \succeq t_j \) for all normal arguments \( t_1, \ldots, t_m \).
   - \( s \succ_{\text{pop}^+} t_j \) for all safe arguments \( t_{m+1}, \ldots, t_{m+n} \).
   - \( t \) contains at most one occurrence of \( f \).
4. \( f \) is recursive, \( t = f(t_1, \ldots, t_{k+1}, \ldots, t_{k+l}) \) and the following conditions hold:
   - \( (s_1, \ldots, s_k) \succ_{\text{pop}^+} (t_1(\overline{t_1}), \ldots, t_{\pi}(\overline{t_{\pi}})) \) for some permutation \( \pi \) on normal argument positions;
   - \( (s_{k+1}, \ldots, s_{k+l}) \succ_{\text{pop}^+} (t_{\pi(1)}, \ldots, t_{\pi(k)}) \) for some permutation \( \pi \) on safe argument positions.

Here \( \succ_{\text{pop}^+} \) denotes the reflexive closure of \( \succ_{\text{pop}^+} \), and we use the orders also for their extension to products of terms.

A TRS \( \mathcal{R} \) is compatible with \( \succ_{\text{pop}^+} \) if all rules are oriented from left to right. We encourage the reader to exercise the definition of \( \succ_{\text{pop}^+} \) by orienting the defining rules of \( B_{\text{vec}} \).

Notice that Clause 3, where we require that the defined symbol \( f \) is recursive, is only needed to orient rules defined by the recursion schema SRN. To measure the recursion depth, we define the depth of recursion \( \text{rd}(f) \) of a function symbol \( f \) defined based on the precedence \( \succ \) as follows: \( \text{rd}(f) := 1 + d \) if \( f \) is recursive; otherwise \( \text{rd}(f) := d \) where \( d = \max\{\text{rd}(g) \mid \ f \succ g \} \). We say a constructor TRS \( \mathcal{R} \) is predicative recursive of degree \( d \) if \( \mathcal{R} \) is compatible with an instance \( \succ_{\text{pop}^+} \) and the maximal depth of recursion of a function symbol in \( \mathcal{R} \) is \( d \).

**Theorem 3.** If \( \mathcal{R} \) is predicative recursive of degree \( d \), then the innermost runtime complexity of \( \mathcal{R} \) lies in \( O(n^d) \).

By the invariance theorem \([2]\) predicative recursive TRSs thus define functions from FP. The converse direction can also be shown. Below the witnessing TRS \( \mathcal{R}_f \) is obtained via a formulation of the class \( B_{\text{vec}} \) from Fig. 1 as term rewriting characterisation of the class \( B_{\text{vec}} \) in the spirit of \([14]\).

**Theorem 4.** For every \( f \in \text{FP} \) there exists a predicative recursive TRS \( \mathcal{R}_f \) computing \( f \).

We obtain that sPOP* characterises FP. sPOP* has been implemented, experiments are available online\(^2\). Initial results also show that integrating sPOP* in \( \text{TCF} \) results in a handful of systems where a tight bound could not be proven before.

REFERENCES


---

**Initial Functions**

\[ P(\cdot; \epsilon) = \epsilon \]

\[ P(S_i(x); x) = x(i = 0, 1) \]

\[ O(\overline{\epsilon}; \overline{y}) = \epsilon \]

**Weak Composition (WSC)**

\[ f(\overline{y}; \overline{g}) = h(x_1, \ldots, x_{ik}; \overline{g}(\overline{x}; \overline{y})) \]

**Safe Recursion (SRN)**

\[ f(\epsilon; \overline{x}; \overline{y}) = g(\overline{x}; \overline{y}) \]

\[ f(S_i(x); \overline{x}; \overline{y}) = h_i(x; \overline{x}; \overline{y}, f(x; \overline{x}; \overline{y}))(i = 0, 1) \]

Fig. 1. Defining initial functions and operations for \( B_{\text{vec}} \). Binary words are constructed from \( \epsilon \) and the dyadic successors \( S_0 \) and \( S_1 \).
Views: Compositional Reasoning for Concurrency

Thomas Dinsdale-Young†, Lars Birkedal*, Philippa Gardner†, Matthew Parkinson†, Hongseok Yang§
†Imperial College London, *IT University of Copenhagen, ‡Microsoft Research Cambridge, §University of Oxford

Abstract—We present a framework for reasoning compositionally about concurrent programs. At its core is the notion of a view: an abstraction of the state that takes account of the possible interference due to other threads. Threads’ views are composable, and an update to the state by one thread must preserve the views of other threads. Existing and novel concurrency reasoning systems can be expressed as instances of our framework, and can be proved sound by appeal to a general soundness result.

I. INTRODUCTION

There has been a recent flurry of research activity on type systems and program logics for reasoning modularly about programs with dynamically allocated, shared mutable state. Type systems have been extended with linear types and related capability system that enforce a mixture of local and global properties. Program logics, extending separation logic, have been developed to reason about various notions of sharing: for sequential languages, by combining with types and various frame rules; for concurrent languages by adding invariant or relational reasoning.

These developments have led to increasingly elaborate reasoning systems, each introducing new features to tackle specific applications of modular reasoning and new metatheory to justify these features. Despite their ad hoc nature, these systems employ a common approach to compositionality. They provide thread-specific abstractions of the state, which embody enough information to prove the behaviour of a thread whilst allowing for the possible behaviours of other threads. We introduce a general framework for these abstractions, identifying the properties necessary for sound, compositional reasoning.

Our fundamental idea is that threads have different views of the machine. Intuitively, a thread’s view consists of information about the current state of the machine, the right of the thread to modify the state as long as the environment’s view is stable (invariant) with respect to such changes, and the thread’s right to the stability of its own view with respect to changes being made by the environment. Threads’ views can be composed, which ensures that the rights and information held by different threads are compatible with each other.

A thread’s view provides a partial, abstract description of the state of the machine. It is partial in that it only describes the state relevant to the thread. It is abstract in that the verifier can use additional information to help with the reasoning, such as types, ghost state or permissions. Such instrumentation has no representation in the concrete state but is a useful fiction for the verifier. To relate the program logic with the operational semantics, we require that a view can be reified to a set of concrete machine states. Using reification, we prove a general soundness result, which we have formally verified in Coq.

To illustrate the essential compositionality of views, consider a command $C$ that updates the view $p$ to the view $q$. For compositional reasoning, we would require that $C$ updates $p \ast r$ to $q \ast r$, where $r$ represents any view held by the environment and $\ast$ is the composition operation on views. Traditional approaches in separation logic have achieved this by enforcing that commands satisfy so-called locality conditions [1]. We take the alternative approach of embedding compositionality into the meaning of “$C$ updates $p$ to $q$”: for all $r$, it must update $p \ast r$ to $q \ast r$. We show that this interpretation, which has been used for formulating separation logics for concurrency and for higher-order languages, gives a simpler and more general metatheory for logics for concurrent programs.

A crucial implication of this interpretation is that views should be stable with respect to any operation that a thread with a consistent view could perform. At one extreme, stability can be enforced by disjointness between views: one thread can access variable $x$, say, while the other cannot have anything to do with $x$. At the other, stability can be enforced by invariant properties: both threads may agree that $x$ always has type bool, for instance. In the middle ground lie many logics that allow controlled sharing. Views capture this whole spectrum.

In this middle ground, the logic of concurrent abstract predicates [2] (CAP) models interference with rely and guarantee relations, which pervade the soundness proof. The rely relation is used to describe the interference that a thread must expect from the environment, and assertions are required to be stable (invariant) under the rely. The guarantee relation constrains (somewhat conservatively) the possible updates that a thread may make, to ensure that it cannot do anything the environment does not expect. In the views framework, we simply take views to be the stable assertions. The guarantee constraint is enforced by the embedding of compositionality in the semantics, which permits any operation that the environment expects. In Kripke models of type theories, the future world relation plays a similar role to the rely in describing possible updates, and is treated similarly in the views framework.

The views framework distills the essence of compositional reasoning. In [3], we consider a range of examples as instantiations of our framework, including simple type systems and separation logics, type systems with recursive types and unique references, a combination of separation logic and a type system in the spirit of [4], and CAP [2].

The framework is already being used to develop logics for advanced language features. CAP has been extended with higher-order features and the soundness of this uses the views framework extended with step-indexing [5]. Views have been extended to reason about $C^\ast$ with interesting permission
The proof rules are standard rules from disjoint concurrent separation logic. They include the frame rule, which captures the intuition that a program’s view can be extended with a composable view, and the disjoint concurrency rule, which allows the views of two threads to be composed.

The operational semantics is parameterised by a model of machine states and an interpretation of the atomic commands as state transformers.

**Parameter D** (Machine States). Assume a set of machine states $S$, ranged over by $s$.

**Parameter E** (Interpretation of Atomic Commands). Assume a function $[-] : \text{Atom} \to S \to P(S)$ that associates each atomic command with a non-deterministic state transformer.

From machine state $s$, the set of states $[a](s)$ is the set of possible outcomes of running the atomic command $a$. If the set is empty, then the command blocks. Here, we consider partial correctness, and so ignore blocking executions.

The operational semantics of the language is expressed by multi-step transition relation $\vdash \rightarrow^* \vdash$ : $(\text{Comm} \times S) \times (\text{Comm} \times S)$. We omit the definition here.

To prove the program logic sound with respect to the operational semantics, we must relate the views (partial, abstract states) with the machine states (concrete, complete states).

**Parameter F** (Reification). Assume a reification function $[-] : \text{View} \to P(S)$, mapping views to sets of machine states.

Soundness requires that the axioms concerning atomic commands are satisfied by the operational interpretation of the commands. For each axiom $(p, a, q)$, the interpretation of $a$ must update view $p$ to $q$ while preserving any environment view. This is captured by the following property:

**Property G** (Atomic Soundness). For every $(p, a, q) \in \text{Axiom}$, and every $r \in \text{View}$ then $[a][p \ast r] \subseteq [q \ast r]$.

This property is both necessary and sufficient for the soundness of the program logic. We state the soundness result here; proof details, including Coq proof scripts, are available [3].

**Theorem 1** (Soundness). Assume that $\vdash \{p\} C \{q\}$ is derivable in the program logic. Then, for all $s \in [p]$ and $s' \in S$, if $(C, s) \rightarrow^* (\text{skip}, s')$ then $s' \in [q]$.

REFERENCES


The constructive content of proofs has always been a central topic of proof theory and it is also one of the most important influences that logic has on computer science. Classical logic is widely used and presents interesting challenges when it comes to understanding the constructive content of its proofs. These challenges have therefore attracted considerable attention, see for example [1][2][3], [4], [5][6], [7], [8] or [9] for different investigations in this direction.

A well-known, but not yet well-understood, phenomenon is that a single classical proof usually allows several different constructive readings. From the point of view of applications this means that we have a choice among different programs that can be extracted. In [10] the authors show that two different extraction methods applied to the same proof produce two programs, one of polynomial and one of exponential average-case complexity. This phenomenon is further exemplified by case studies in [5], [11], [12] as well as the asymptotic results [13], [14]. The reason for this behavior is that classical “proofs often leave algorithmic detail underspecified” [15].

On the level of cut-elimination in the sequent calculus this phenomenon is reflected by the fact that the standard proof reduction without imposing any strategy is not confluent. In this paper we consider cut-elimination in classical first-order logic and treat the question which cut-free proofs one can obtain (by the strategy-free rewriting system) from a single proof with cuts. As our aim is to compare cut-free proofs obtained from the compatible and transitive closure of the standard set of local reduction rules: inference permutations, duplication on eliminatin of weakening, and the reductions of axioms and logical connectives. We write \( \cdot \) for \( \cdot \) without the reduction rule for weakening.

Note that in general a proofs is a Herbrand-disjunction. Non-erasing reduction is for the cut-reduction relation on proofs obtained from the compatible and transitive closure of the standard set of local reduction rules: inference permutations, duplication on elimination of contraction, deletion on elimination of weakening, and the reductions of axioms and logical connectives. We write \( \cdot \) for \( \cdot \) without the reduction rule for weakening.

Definition. Let \( A \) be quantifier-free, let \( \pi \) be a cut-free proof of \( \exists x A \) and let \( t_1, \ldots, t_n \) be all terms used for instantiating \( \exists x A \) in \( \pi \). The Herbrand-set of \( \pi \) is defined as \( H(\pi) = \{ A[x\setminus t_1], \ldots, A[x\setminus t_n] \} \).

The formula \( \bigvee_{A \in H(\pi)} A \) is a tautology and called Herbrand-disjunction of \( \pi \).

Previous work [16], [17] has established a connection between proof theory and formal language theory on which we base the work of this paper. We use rigid tree languages which have been introduced in [18] with applications in verification (e.g. of cryptographic protocols as in [19]) as their primary purpose. To a proof \( \pi \) we associate a rigid tree grammar \( G(\pi) \) and via this grammar define the Herbrand-content of \( \pi \) as \( [\pi] = L(G(\pi)) \), the language induced by \( \pi \)’s grammar. The grammar \( G(\pi) \) essentially describes the dependencies between the quantifiers in the cut formulas of \( \pi \). For cut-free proofs \( [\pi] = H(\pi) \).

We restrict our attention to proofs whose cut formulas are of the form \( \exists x C \) for \( C \) quantifier-free s.t. the \( \exists \)-inference introducing this quantifier is immediately above the cut. A proof not fulfilling the condition on the position of the quantifier introduction can be easily pruned into one that does by shifting rules and identifying eigenvariables. Such proofs will be called simple in the following.

We work in a multiplicative version of the sequent calculus and write \( \cdot \) for the cut-reduction relation on proofs obtained from the compatible and transitive closure of the standard set of local reduction rules: inference permutations, duplication on elimination of contraction, deletion on elimination of weakening, and the reductions of axioms and logical connectives. We write \( \cdot \) for \( \cdot \) without the reduction rule for weakening.

Note that a \( \cdot \) is an analytic proof as well, e.g. \( H(\pi) \) is a Herbrand-disjunction. Non-erasing reduction is frequently studied in the context of the lambda-calculus, often in the form of the \( \lambda \)-calculus where it gives rise to the conservation theorem (see Theorem 13.4.12 in [20]). Our situation here is however quite different: neither \( \cdot \) nor \( \cdot \) is confluent and neither of them is strongly normalizing. Nevertheless we obtain:

Theorem. If \( \pi \) \( \cdot \) \( \pi' \) is a reduction sequence of simple proofs, then \( [\pi] \subseteq [\pi'] \). If \( \pi \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \) of simple proofs, then \( [\pi] = [\pi'] \).

Definition (Herbrand-confluence). A relation \( \cdot \) on a set of proofs is called Herbrand-confluent iff \( \cdot \pi_1 \pi_2 \) with \( \pi_1 \) and \( \pi_2 \) being normal forms for \( \cdot \) implies that \( H(\pi_1) = H(\pi_2) \).
Corollary. The relation $\rightsquigarrow$ is Herbrand-confluent on the set of simple proofs.

The central proof technique is the utilization of tree grammars and modification of derivations in such grammars. This strong connection to formal language theory has the side effect that standard operations on formal languages such as for example membership or intersection, assume a proof-theoretic meaning by allowing to decide whether a given term is witness of a proof, or, respectively, by characterizing the set of witnesses obtainable from both of two given proofs, etc. Another side effect of this proof technique is a combinatorial description of how the structure of a cut-free proof is related to that of a proof with cut. Such descriptions are important theoretical results which underlie applications such as algorithmic cut-introduction, see [21].

How do these results fit together with $\rightsquigarrow$ being neither confluent nor strongly normalizing? In fact, note that it is possible to construct a simple proof which permits an infinite $\rightsquigarrow$ reduction sequence from which one can obtain normal forms of arbitrary size by bailing out from time to time. This can be done by building on the propositional double-contraction example found e.g. in [2], [22], [5] and in a similar form in [23]. While these infinitely many normal forms do have pairwise different Herbrand-disjunctions when regarded as multisets, the above corollary shows that as sets they are all the same. This observation shows that the lack of strong normalization is taken care of by using sets instead of multisets as data structure. But what about the lack of confluence? Results like [13] and [14] show that the number of $\rightsquigarrow$ normal forms with different Herbrand-disjunctions can be enormous. On the other hand we have just seen that $\rightsquigarrow$ induces only a single Herbrand-disjunction: $\llbracket \pi \rrbracket$. The relation between $\llbracket \pi \rrbracket$ and the many Herbrand-disjunctions induced by $\rightsquigarrow$ is explained by the first part of the above theorem: $\llbracket \pi \rrbracket$ contains them all as subsets.

Given the wealth of different methods for the extraction of constructive content from classical proofs, what we learn from our work is this: the first-order structure possesses (in contrast to the propositional structure) a unique and canonical unfolding. The various extraction methods hence do not differ in the choice of how to unfold the first-order structure but only in choosing which part of it to unfold. We therefore see that the effect of the underspecification of algorithmic detail in classical logic is redundancy.

As future work, the authors plan to extend this result to arbitrary first-order proofs. Stronger classes of cut formulas require correspondingly stronger classes of tree grammars. Concerning further generalizations, note that the method of describing a cut-free proof by a tree language is applicable to any proof system with quantifiers that has a Herbrand-like theorem, e.g., even full higher-order logic as in [24]. The difficulty consists in finding an appropriate type of grammars. The reader interested in more details is referred to the full version at http://www.logic.at/people/hetzl/hcon.pdf.

REFERENCES

The Refined Calculus of Inductive Construction: Parametricity and Abstraction

Chantal Keller
INRIA Saclay–Île-de-France at École Polytechnique
Email: Chantal.Keller@inria.fr

Marc Lasson
ENS Lyon, Université de Lyon, LIP
UMR 5668 CNRS ENS Lyon UCBL INRIA
Email: marc.lasson@ens-lyon.org

Abstract—We present a refinement of the Calculus of Inductive Constructions in which one can easily define a notion of relational parametricity. It provides a new way to automate proofs in an interactive theorem prover like Coq.

I. INTRODUCTION

The Calculus of Inductive Constructions (CIC in short) extends the Calculus of Constructions with inductively defined types. It is the underlying formal language of the Coq interactive theorem prover [1].

In the original presentation, CIC had three kinds of sorts: the impredicative sort of propositions Prop, the impredicative sort of basic informative types Set, and the hierarchy of universes Type₀, Type₁, ... This presentation was not compatible with the possibility to add axioms in the system, since it could lead to inconsistencies [2]. Nowadays, there is no impredicative sort of basic informative types, and Set represents Type₀.

This does not fit well with one of the major original ideas about CIC: the possibility to perform program extraction. Indeed, since the current version of CIC does not separate informative types from non-informative types, extraction needs to normalize its type to guess whether it should be erased or not, and this makes it very uneasy to prove correct [3].

In this paper, we propose a refinement of CIC which reconciles extraction with the possibility to add axioms in the system: CIC_ref, the Refined Calculus of Inductive Constructions. The idea is to split the (Typeᵢ)ᵢ∈ℕ hierarchy into two hierarchies (Setᵢ)ᵢ∈ℕ and (Typeᵢ)ᵢ∈ℕ⁺, one for informative types and one for types without computational content.

This calculus allows us to extend the presentation of parametricity for Pure Types Systems introduced by Bernardy et al. [4] to the Calculus of Inductive Constructions. Parametricity is a concept introduced by Reynolds [5] to study the type abstraction of system F, and the abstraction theorem expresses the fact that polymorphic programs map related arguments to related results. In CIC_ref, we can define a notion of relational parametricity in which the relations’ codomains is the Prop sort of propositions.

II. CIC_ref: THE REFINED CALCULUS OF INDUCTIVE CONSTRUCTIONS

The Refined Calculus of Inductive Constructions is a refinement of CIC where terms are generated by the same grammar as CIC:

\[ A, B, P, Q, F ::= x \mid s \mid \forall x : A.B \mid \lambda x : A.B \mid (A) \mid I \mid \text{case}_{i}(A, Q, P, F) \mid c \mid \text{fix}(x : A).B \]

where s ranges over the set \{Prop\} \cup \{Setᵢ, Typeᵢ₊₁ | i ∈ N\} of sorts and x ranges over the set of variables. We write Indᵢ(\text{I}: A, c : C) to state that I is a well-formed inductive definition typed with p parameters, of arity A, with k constructors c₁, ..., cᵦ of respective types C₁, ..., Cᵦ.

A context Γ is a list of pairs x : A and the typing rules are the rules of CIC (one can refer to [1] for the complete set of rules), except to type sorts and dependent products. As for CIC, typing fixpoints (for \text{fix}) and elimination rules (for \text{case}) is subject to restrictions to ensure coherence. We present only the rules which are specific to our type system. Here are the three typing rules to type sorts:

\[ \Gamma \vdash \text{Prop} : \text{Type}_1 \]
\[ \Gamma \vdash \text{Set}_i : \text{Type}_{i+1} \]
\[ \Gamma \vdash \text{Type}_{i} : \text{Type}_{i+1} \]

The following three typing rules tell which products are authorized in the system. The level of the product is the maximum level of the domain and the codomain:

\[ \Gamma \vdash A : r_i \quad \Gamma, x : A \vdash B : s_j \quad (r, s) \in \{\text{Type}, \text{Set}\} \]

\[ \Gamma \vdash \forall x : A.B : s_{\text{max}(i,j)} \]

Quantifying over propositions does not rise the level of the product:

\[ \Gamma \vdash A : \text{Prop} \quad \Gamma, h : A \vdash B : s_i \quad s \in \{\text{Set}, \text{Prop}\} \]

And the sort Prop is impredicative, it means that products in Prop may be built by quantifying over objects whose types inhabit any sort:

\[ \Gamma \vdash A : s \quad \Gamma, x : A \vdash B : \text{Prop} \quad \Gamma \vdash \forall x : A.B : \text{Prop} \quad s \in \{\text{Set}, \text{Prop}\} \]

Finally, as in CIC, the system comes with subtyping rules based on the following inclusion of sorts (where i < j):

\[ \text{Prop} :\text{Set}_i \quad \text{Set}_i :\text{Set}_j \quad \text{Type}_i :\text{Type}_j \]

One should note that CIC_ref easily embeds into CIC by mapping any Setᵢ and Typeᵢ onto the Typeᵢ of CIC. The coherence of CIC thus implies the coherence of CIC_ref.

III. PARAMETRICITY

We can define a notion of relational parametricity for CIC_ref.
\[ \Theta_I(Q^P, T, F^n) = \lambda(x : A)(x' : A')(x_R : \Gamma \xrightarrow{A} x'_{\Gamma}) (a : I Q^P \xrightarrow{x^n} (a' : I Q^P \xrightarrow{x^n}) (a_R : I) Q Q' \xrightarrow{Q'} x'_{\Gamma} x_R a' a). \]

\[ [T] x'_{\Gamma} x_R a' a_R (\text{case}_I(a, Q^P, T, F^n)) \text{ (case}_I(a', Q^P, T', F^n)) \]

Fig. 1. Relation parametricity for inductive types

**Definition 1** (Parametricity relation). For any inductive
\[ \text{In}_d(I : A, c : C^k), \text{ we define a fresh inductive symbol } [I] \]
and a family \((\Gamma_i)_{i=1\ldots k}\) of fresh constructor names.

The parametricity translation \(\llbracket \cdot \rrbracket\) is defined by induction on the structure of terms and contexts:

\[ \llbracket () \rrbracket = () \]
\[ \llbracket \Gamma, x : A \rrbracket = \llbracket \Gamma \rrbracket, x : A, x' : A', x_R : [A] x' \]
\[ \llbracket \text{s} \rrbracket = \lambda(x : s)(x' : s). x \rightarrow x' \rightarrow s \]
\[ \llbracket [x] \rrbracket = x_R \]
\[ \llbracket \forall x : A.B \rrbracket = \lambda(f : \forall x : A.B)(f' : \forall x' : A'.B'). \forall(x : A)(x' : A')(x_R : [A] x'). \llbracket B \rrbracket (f(x')) \]
\[ \llbracket \lambda x : A.B \rrbracket = \lambda(x : A)(x' : A')(x_R : [A] x'). \llbracket B \rrbracket \]
\[ \llbracket \text{fix}(x : A).B \rrbracket = \text{fix}(x_R : [A] x'). \llbracket B \rrbracket \]
\[ \llbracket \text{case}_I(M, Q^P, T, F^n) \rrbracket = \text{case}_I(M, Q, Q', [Q]^p, \Theta_I(Q^P, T, F^n), \llbracket \text{case}_I(a, Q^P, T, F^n) \rrbracket) \]

where \(\text{Prop} = \text{Set}_i = \text{Prop} \text{ and Type}_i = \text{Type}_i\), and where \(\text{A'}\) denotes the term \(A\) in which we have replaced each variable \(x\) by a fresh variable \(x'\). The definition of \(\Theta\) is in Fig. 1.

What is new with respect to previous works is the fact that relations over objects of type \(\text{Prop}\) or \(\text{Set}_i\) have their codomain in \(\text{Prop}\) instead of higher universes. We also formally define parametricity for inductive types.

Unfortunately, in order to prove the abstraction theorem below, we need to restrict the strong elimination: we have to disallow the case destructors used to build objects whose types are of sort \(\text{Type}\) when the destructed inductive definition is not small (small inductive definitions are inductive definitions which constructors only have arguments of type \(\text{Prop}\) or \(\text{Set}_i\), see [6]). We write \(\vdash_{\text{f}}\) for the derivability where strong elimination is authorized only over small inductive definitions.

**Theorem 1** (Abstraction theorem). If \(\Gamma \vdash_{\text{f}} A : B\) then \(\llbracket \Gamma \rrbracket \vdash_{\text{f}} [A] : [B] \rightarrow [A'] : [B']\).

**IV. Applications**

A lot of so-called “free theorems” are consequences of the abstraction theorem and our framework is expressive enough to implement most examples that can be found in the literature (see for instance [4], [7]).

Here we propose a new example inspired by François Garillot’s thesis [8], in which he remarks that polymorphic functions operating on groups can only compose elements using the laws given by the group’s structure, and thus cannot create new elements.

In our system, we may actually use parametricity theory to translate this uniformity property. We take an arbitrary group structure \(\mathcal{H}\) defined by its carrier \(\alpha : \Sigma\), a unit element, a composition law, an inverse and the standard axioms stating that \(\mathcal{H}\) is a group. We define \(\text{fingrp}\) the type of all the finite subgroups of \(\mathcal{H}\) consisting of a list plus stability axioms. Now consider any term \(Z : \text{fingrp} \rightarrow \text{fingrp}\) (examples of such terms abound: e.g. the center, the normalizer, the derived subgroup...). The abstraction theorem states that for any \(R : \alpha \rightarrow \alpha \rightarrow \text{Prop}\) compatible with the laws of \(\mathcal{H}\) and for any \(G G' : \text{fingrp}\), \(\llbracket \text{fingrp} \rrbracket R(G, G') = \llbracket \text{fingrp} \rrbracket R(Z G, Z G')\) where \(\llbracket \text{fingrp} \rrbracket R\) is the relation on subgroups induced by \(R\).

Given this, we can prove the following properties:

- for any \(G, Z G \subset G\) (if we take \(R : x y \rightarrow x \in G\));
- for any \(G, \) for any \(\phi\) a morphism of \(\mathcal{H}\), \(\phi(Z G) = Z \phi(G)\) (if we take \(R : x y \rightarrow y = \phi(x)\)).

It entails that \(Z G\) is a characteristic subgroup of \(\mathcal{H}\).

For a complete Coq formalization of this, please refer to the online source code [9].

**V. Conclusion**

The system presented here allows to distinguish clearly via typing which expressions will be computationally meaningful after extraction. It allows us to define a notion of parametricity for which relations lie in the sort of propositions. We set here the theoretical foundation for an implementation of a Coq tactic that constructs proof terms by parametricity. A first prototype of such a tactic can be found online [9].

**REFERENCES**

Undecidable First-Order Theories of Affine Geometries

Antti Kuusisto
University of Tampere
anti.j.kuusisto@uta.fi

Jeremy Meyers
Stanford University
jmeyers@stanford.edu

Jonni Virtema
University of Tampere
jonni.virtema@uta.fi

I. INTRODUCTION

Tarski initiated a logic-based approach to formal geometry that studies first-order structures with a ternary betweenness relation (β) and a quaternary equidistance relation (≡). Tarski established, inter alia, that the first-order (FO) theory of (\(\mathbb{R}^2, \beta, \equiv\)) is decidable. For further information on the development of Tarski’s geometry, see [11]. Aiello and van Benthem conjectured in [1] that the FO-theory of the class of expansions of (\(\mathbb{R}^2, \beta\)) by unary predicates is decidable. We refute this conjecture by showing that for all \(n \geq 2\), the FO-theory of the class of monadic expansions of (\(\mathbb{R}^n, \beta\)) is \(\Pi_1^1\)-hard and therefore not even arithmetical. We also define a natural and comprehensive class \(\mathcal{C}\) of geometric structures (\(T, \beta\)), where \(T \subseteq \mathbb{R}^n\), and show that the for each structure (\(T, \beta\)) \(\in \mathcal{C}\), the FO-theory of the class of monadic expansions of (\(T, \beta\)) is undecidable. We then consider classes of expansions of structures (\(T, \beta\)) with restricted unary predicates, for example finite predicates, and establish a variety of related undecidability results. In addition to decidability questions, we briefly study the expressivity of universal MSO and weak universal MSO over expansions of (\(\mathbb{R}^n, \beta\)). While the logics are incomparable in general, over expansions of (\(\mathbb{R}^n, \beta\)), formulae of weak universal MSO translate into equivalent formulae of universal MSO.

Our results could turn out interesting in investigations concerning logical aspects of spatial databases. It turns out that there is a canonical correspondence between (\(\mathbb{R}^2, \beta\)) and (\(\mathbb{R}, 0, 1, +, <\)), see [7]. See the survey [9] for further details on logical aspects of spatial databases.

The betweenness predicate is also studied in spatial logic [3]. The recent years have witnessed a significant increase in the research on spatially motivated logics. Several interesting systems with varying motivations have been investigated, see the surveys [2] and [4]. Our results contribute to the understanding of spatially motivated first-order languages, and hence they can be useful in the search for decidable (modal) spatial logics.

II. PRELIMINARIES

Tiling methods constitute a flexible framework for establishing different degrees of undecidability of different kinds of problems. An input to a tiling problem is a finite set of tile types, i.e., a finite set of rectangles with coloured edges. The problem is to decide whether it is possible to tile a predetermined region of space with tiles of the given type, under the constraint that adjacent edges of tiles have the same colour. We make use of the three following variants of the tiling problem. The standard tiling problem asks whether a set \(T\) of tile types can tile the \(\mathbb{N} \times \mathbb{N}\) grid. The recurrent tiling problem asks whether \(T\) and some assigned tile type \(t \in T\) can tile the \(\mathbb{N} \times \mathbb{N}\) grid such that \(t\) occurs infinitely many times on the leftmost column of the grid, and the torus tiling problem asks if there exists some finite torus (i.e., a finite grid whose borders wrap around to form a torus) such that the input set \(T\) tiles it.

Theorem II.1. The tiling problem is \(\Pi_1^1\)-complete [5], the recurrent tiling problem \(\Sigma_1^1\)-complete [8], and the periodic tiling problem \(\Sigma_1^1\)-complete [6].

Let (\(\mathbb{R}^n, d\)) be the \(n\)-dimensional Euclidean space with the canonical metric \(d\). We define the ternary Euclidean betweenness relation \(\beta\) such that \(\beta(s, t, u)\) iff \(d(s, u) = d(s, t) + d(t, u)\). We study geometric betweenness structures of the type (\(T, \beta\)), where \(T \subseteq \mathbb{R}^n\) and where \(\beta\) is the restriction of the betweenness predicate of \(\mathbb{R}^n\) to the set \(T\).

A subset \(S \subseteq \mathbb{R}^n\) is an \(m\)-dimensional flat of \(\mathbb{R}^n\), where \(0 \leq m \leq n\), if there exists a set of \(m\) linearly independent vectors \(v_1, \ldots, v_m \in \mathbb{R}^n\) and a vector \(h \in \mathbb{R}^n\) such that \(S\) is the \(h\)-translated span of the vectors \(v_1, \ldots, v_m\), in other words \(S = \{u \in \mathbb{R}^n | u = h + r_1v_1 + \ldots + r_mv_m, r_1, \ldots, r_m \in \mathbb{R}\}\).

Note that \((0, \ldots, 0)\) is not considered to be a linearly independent set.

A set \(U \subseteq \mathbb{R}^n\) is a linearly regular \(m\)-dimensional flat, where \(0 \leq m \leq n\), if the following conditions hold.

1) There exists an \(m\)-dimensional flat \(S\) such that \(U \subseteq S\).
2) There does not exist any \((m - 1)\)-dimensional flat \(S\) such that \(U \subseteq S\).
3) \(U\) is linearly complete, i.e., if \(L \subseteq U\) is a line in \(U\) and \(L' \supseteq L\) the corresponding line in \(\mathbb{R}^n\), and if \(r \in L'\) is a point and \(\epsilon \in \mathbb{R}_+\), a positive real number, then there exists a point \(s \in L\) such that \(d(s, r) < \epsilon\). Here \(d\) is the canonical metric of \(\mathbb{R}^n\).
4) \(U\) is linearly closed, i.e., if points \(x_1, x_2 \in U\) determine two lines that intersect in \(\mathbb{R}^n\), then the corresponding lines in \(U\) intersect in \(U\).

A set \(T \subseteq \mathbb{R}^n\) extends linearly in \(mD\), where \(m \leq n\), if there exists a linearly regular \(m\)-dimensional flat \(S\), a positive real number \(\epsilon \in \mathbb{R}_+\), and a point \(x \in S \cap T\) such that \(\{u \in \mathbb{R}^n | u = h + r_1v_1 + \ldots + r_mv_m, r_1, \ldots, r_m \in \mathbb{R}\})
S \mid d(x, u) < \epsilon \} \subseteq T. It is easy show that for example the rational plane $\mathbb{Q}^2$ and the closed rectangle $[0, 1] \times [0, 1] \subseteq \mathbb{R}^2$ extend linearly in $2D$.

### III. Results

While $\forall\text{WMSO} \not\subseteq \text{MSO}$ and $\forall\text{MSO} \not\subseteq \text{WMSO}$ in general, over models embedded in $\langle \mathbb{R}^n, \beta \rangle$, $\forall\text{WMSO}$ translates into $\forall\text{MSO}$ and $\forall\text{WMSO}$ into $\forall\text{MSO}$.

**Theorem III.1** (Heine-Borel). A set $S \subseteq \mathbb{R}^n$ is closed and bounded iff every open cover of $S$ has a finite subcover.

**Theorem III.2.** Let $C$ be the class of expansions $\langle \mathbb{R}^n, \beta, P \rangle$ of $\langle \mathbb{R}^n, \beta \rangle$ with a unary predicate $P$, and let $F \subseteq C$ be the subclass of $C$ where $P$ is finite. The class $F$ is first-order definable with respect to $C$.

**Proof:** It follows directly from the Heine-Borel theorem that a set $T \subseteq \mathbb{R}^n$ is finite iff it is closed, bounded and consists of isolated points of $T$. The proof of the current theorem relies on this fact. The argument is based on encoding topological information about open balls by first-order formulae. The idea is to replace open balls by open $n$-dimensional triangles.

We first define a formula $\text{par}(x, y, u, v)$ stating in $\mathbb{R}^n$ that the lines defined by $x, y$ and $u, v$ are parallel. With this formula we construct formulae $\text{basis}_k(x_0, \ldots, x_k)$ and $\text{flat}_k(x_0, \ldots, x_k, z)$ by simultaneous recursion. The formulae state roughly that vectors $(x_0, x_i)$ form a basis of $\mathbb{R}^k$-dimensional flat, and that $z$ is in the flat. With these formulae we recursively define formulae $\text{opentriangle}(x_0, \ldots, x_k, z)$ stating that $z$ is in the $k$-dimensional open triangle defined by the points $x_0, \ldots, x_k$.

The first-order theory of the class of expansions $\langle T, \beta, P_{i \in \mathbb{N}} \rangle$ of any structure $\langle T, \beta \rangle$ that extends linearly in $2D$ is undecidable. Here $P_i$ are monadic predicates. This is shown by interpreting the $\mathbb{N} \times \mathbb{N}$ grid, or some superstructure of the grid, in the class of monadic expansions of $\langle T, \beta \rangle$. This is done by defining two linear sequences of points that are stored in two dimensional open triangle defined by the points $x_0, \ldots, x_k$.

---

**Theorem III.3.** Let $T \subseteq \mathbb{R}^n$ be a set that extends linearly in $2D$. The monadic $\Pi_1$-theory of $\langle T, \beta \rangle$ is $\Sigma_1^n$-hard.

Extending linearly in $1D$ is not a sufficient condition for undecidability of the $\forall\text{MSO}$-theory of $\langle T, \beta \rangle$. This can be seen from the fact that the $\forall\text{MSO}$-theory of $\langle \mathbb{Q}, < \rangle$ is decidable [10].

In $\mathbb{R}^n$, it is possible to define predicates $P$ and $Q$ such that they correspond to linear sequences of the order type $\omega$ exactly. This enables the encoding of an isomorphic copy of the $\mathbb{N} \times \mathbb{N}$ grid in expansions of structures $\langle \mathbb{R}^n, \beta \rangle$, where $n \geq 2$. With an isomorphic copy of the grid, we can interpret the recurring tiling problem in the structure created.

**Theorem III.4.** Let $n \geq 2$. The monadic $\Pi_1$-theory of the structure $\langle \mathbb{R}^n, \beta \rangle$ is $\Pi_1^{\text{ar}}$-hard, and therefore not even arithmetical.

When limiting attention to expansions of structures $\langle T, \beta \rangle$ with finite monadic predicates, we can use the periodic tiling problem in order to establish undecidability of monadic expansion classes of structures $\langle T, \beta \rangle$.

**Theorem III.5.** Let $T \subseteq \mathbb{R}^n$ be a set that extends linearly in $2D$. The weak monadic $\Pi_1$-theory of $\langle T, \beta \rangle$ is $\Pi_1^{\text{ar}}$-hard.

In addition to expansions with finite predicates, the periodic tiling problem can be easily modified to yield undecidability of a wide variety of natural restricted monadic expansions classes of $\langle \mathbb{R}^2, \beta \rangle$. These include expansions with predicates corresponding to finite unions of closed rectangles, polygons, and other simple canonical classes of sets.

In the future we shall identify fragments of first-order logic that lead to a decidable theory of the monadic expansion class of $\langle \mathbb{R}^2, \beta \rangle$.

---

**REFERENCES**


Abstract—A progress measure for a parity game is a labeling $s(\cdot)$ of the vertices of the game that witnesses winning strategies for the players: Player 0 (Even) and Player 1 (Odd). We give a natural definition of a canonical progress measure for a parity game $G$ directly, without translating into the $\mu$-calculus. The label $s(u)$ of a vertex $u$ is defined to be the value of an infinitely-played game played on the graph of $G$.

In order to show the existence of these values, we introduce a finitely-played version of the game of duration at most $n(n+1)/2$ moves, where $n$ is the number of vertices. We show that the values for the finitely-played version are the same as the values for the infinitely-played version. This result also implies the existence of optimal strategies for both players (in the infinitely-played version) that use memory of size $O(n^{n(n+1)/2})$.

Without loss of generality we restrict attention to parity games in which Player 0 has a winning strategy from every vertex. We show that Player 0 has a memoryless strategy that ensures the canonical progress measure everywhere. For Player 1, optimal strategies are more complex. We consider the special case of 1-solitaire games, i.e. games where only Player 1 has non-trivial moves. For 1-solitaire games, we show that Player 1 must have memory of size $\Omega(n)$ in order to ensure the canonical progress measure. This lower bound extends, of course, to the general case. Moreover, for 1-solitaire games we show that Player 1 has optimal strategies that use memory of size $O(n)$. For the general case, we do not have a matching upper bound for the size of the memory. We improve upon the upper bound previously stated: Player 1 has optimal strategies with memory of size $O(n^n)$.

Our results imply that the canonical progress measure for a parity game $G$ records optimal strategies for Player 0. The same does not seem to be the case, however, for Player 1. We consider this an indication that verifying canonical progress measures for parity games is not easier than finding them.

I. INTRODUCTION

A parity game involves two players, Player 0 (or Even) and Player 1 (or Odd). It is played on a directed graph whose vertices are labeled with natural numbers called priorities. The vertices are partitioned into those that belong to Player 0 (0-vertices) and those that belong to Player 1 (1-vertices). A play starts at some vertex where a token is placed. At every step, the player who owns the vertex with the token moves the token to a successor vertex. Thus, an infinite sequence of vertices is formed. Player 0 wins the play if the maximum priority that appears infinitely often is even, otherwise Player 1 wins.

Parity games are memorylessly determined, i.e. for every vertex $u$ some player $\sigma$ has a memoryless strategy $f_\sigma$ s.t. every play that starts from $u$ with Player $\sigma$ playing according to $f_\sigma$ is won by Player $\sigma$. The winning region of Player 0 is the set of vertices from which Player 0 has a winning strategy. Solving parity games amounts to finding the winning region of Player 0. By determinacy, the rest of the vertices are the winning region of Player 1. The decision version of the problem is: Given a parity game $G$ and a vertex $u$, does Player 0 have a winning strategy from $u$? Memoryless determinacy implies that the problem lies in $\text{NP} \cap \text{coNP}$ [1]. Jurdziński has shown that the problem is even contained in $\text{UP} \cap \text{coUP}$ [2].

The importance of finding fast algorithms for solving parity games lies in its polynomial-time equivalence to the problem of model checking the $\mu$-calculus [3], [4]. Despite efforts of the community no polynomial-time algorithm is known for solving parity games. One line of research for designing algorithms for parity games involves the notion of progress measure [5]. Progress measures, introduced by Klárulnd and Kozen in [6] where they are called Rabin measures, are annotations of graphs that record progress towards the satisfaction of Rabin conditions. Streett and Emerson used a similar notion, which they called signature [7], to study the $\mu$-calculus.

A progress measure for a parity game is a labelling of the vertices of the game that witnesses winning strategies for the players and hence also the winning regions. The progress measure records progress towards the satisfaction of the parity condition. Walukiewicz considers canonical signature assignments for parity games [8], which are defined by translating the existence of a winning strategy into the $\mu$-calculus and then using the notion of signature by Emerson and Streett [7]. Our definition of the canonical progress measure for a parity game is similar, but does not involve the $\mu$-calculus. The canonical progress measure is unique and records winning strategies for the players that are “good” in the sense of minimizing the progress measure. The progress measure is defined as a labelling $s(\cdot)$ of the vertices so that for a vertex $u$, $s(u)$ is the value of a game of infinite duration. This is well-defined, because $s(u)$ is shown to be both the least outcome that Player 0 can ensure and the greatest outcome that Player 1 can ensure. This result is also relevant to addressing a question raised by Jurdziński in [2]: Are canonical progress measures unique succinct certificates for parity games? We do not resolve this question here, but we believe that our results provide an indication for a negative answer.

We feel that studying canonical progress measures is important in furthering our understanding of parity games. This is because finding the winning regions, winning strategies, as well as finding some progress measure for a parity game are all equally hard problems. Finding the canonical progress measure is at least as hard. It amounts to finding “good” winning strategies in a precise sense.
II. SUMMARY OF RESULTS

We define the outcome of an infinite play won by Player 0 (Player 1) to be a function that maps each odd (even) priority to a natural number. We call such a function a 0-signature (1-signature). Consider the lexicographic ordering of these functions, where larger priorities are more significant. Player 0 wants to minimize the outcome and Player 1 to maximize it. We show that for a vertex \( u \) both players have strategies that ensure the same outcome \( s(u) \) for plays starting from \( u \). In order to show this, we consider a finitely-played version of the game of duration \( \leq n(n+1)/2 \), where \( n \) is the number of vertices. In the finitely-played version, a play ends as soon as a cycle is formed after the first occurrence of the maximum priority that has appeared so far. We say that \( s(u) \) is the value of \( u \). The values for the finitely-played version are the same as the values for the infinitely-played version. We show this fact using a technique similar to the one used by Ehrenfeucht and Mycielski in [9], where a similar result is established for mean-payoff games. A corollary is that in the infinitely-played version the players have optimal strategies that use memory of size at most \( O(n(n+1)/2) \). The canonical progress measure is defined to be the value assignment \( s(\cdot) \).

Without loss of generality we study parity games in which Player 0 has a winning strategy from every vertex. First, we establish that Player 0 has a memoryless strategy \( f_0 \) such that for every vertex \( u \), \( f_0 \) ensures outcome \( \leq s(u) \) from \( u \). We show an even stronger result: The canonical progress measure records all memoryless strategies that ensure the measure from every vertex.

We also study optimal strategies for Player 1. For the simpler case of 1-solitaire games, i.e. games where only Player 1 has non-trivial moves, we show that optimal strategies for Player 1 need memory of size at least \( \Omega(n) \). Moreover, optimal strategies can be constructed that use memory of size at most \( O(n) \). In order to construct optimal strategies we introduce the notions of extended outcome and extended value. These notions formalize the idea that Player 1 tries to maximize the outcome in as few steps as possible.

For general parity games, the linear lower bound for the size of the memory also applies. We do not have a matching upper bound. We improve, however, upon the \( O(n(n+1)/2) \) upper bound we stated previously. We show that Player 1 has optimal strategies that use memory of size at most \( O(n) \). Again, constructing these optimal strategies involves the notions of extended outcome and extended value.

III. CONCLUSION

Let \( G \) be a parity game and \( W_0, W_1 \) be the winning regions of Player 0 and 1 respectively. Our results for optimal strategies (in general games) are summarized in the following diagram:

<table>
<thead>
<tr>
<th>( W_0 )</th>
<th>( W_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal 0-strategy: memoryless</td>
<td>optimal 1-strategy: memoryless</td>
</tr>
<tr>
<td>( \Omega(n) \leq</td>
<td>M</td>
</tr>
</tbody>
</table>

We conjecture that there exist optimal 1-strategies for \( G[W_0] \) (and hence also optimal 0-strategies for \( G[W_1] \)) with memory of size at most \( O(n) \), where \( G[W] \) denotes the restriction of \( G \) to \( W \).

The decision version of the problem of finding the canonical progress measure of a parity game is the following: Given a game \( G \), a vertex \( u \), and a 0-signature \( t \), is it the case that \( s(u) \leq t \)? Call this problem CANONICAL. We have preliminary results showing that if the above conjecture is true, then CANONICAL lies in \( \text{NP} \cap \text{coNP} \).

Let \( G \) be a game in which Player 0 wins from every vertex. We have shown that the canonical progress measure records all possible memoryless 0-strategies that ensure the measure. It does not seem, however, that we can read optimal 1-strategies off from the measure. We take this as an indication that verifying canonical progress measures is not easier than finding them, since we might still need to guess an optimal 1-strategy.

REFERENCES

Down the Borel Hierarchy: Solving Muller Games via Safety Games*

Daniel Neider*, Roman Rabinovich†, and Martin Zimmermann*†

* Lehrstuhl für Informatik 7, RWTH Aachen University, Germany
Email: {neider, zimmermann}@automata.rwth-aachen.de
† Mathematische Grundlagen der Informatik, RWTH Aachen University, Germany
Email: rabinovich@logic.rwth-aachen.de
‡ Institute of Informatics, University of Warsaw, Poland

Abstract—We transform a Muller game with \( n \) vertices into a safety game with \((n!)^3\) vertices whose solution allows to determine the winning regions of the Muller game and to compute a finite-state winning strategy for one player. This yields a novel memory structure and a natural notion of permissive strategies for Muller games. Moreover, we generalize our construction by presenting a new type of game reduction from infinite games to safety games and show its applicability to several other winning conditions.

I. INTRODUCTION

Muller games are a source of interesting and challenging questions in the theory of infinite games. They are expressive enough to describe all \( \omega \)-regular properties. Also, all winning conditions that depend only on the set of vertices visited infinitely often can trivially be reduced to Muller games. Hence, they subsume Büchi, co-Büchi, parity, Rabin, and Streett conditions. Furthermore, Muller games are not positionally determined, i.e., both players need memory to implement their winning strategies. In this work, we present a framework to deal with three aspects of Muller games: solution algorithms, memory structures, and quality measures for strategies.

While investigating the interest of Muller games for “casual living-room recreation” [1], McNaughton introduced scoring functions which describe the progress a player is making during a play: consider a Muller game \((A, F_0, F_1)\), where \( A \) is the arena and \((F_0, F_1)\) is a partition of the set of loops in \( A \) used to determine the winner: Player \( i \) wins a play \( \rho \) if the set of vertices visited infinitely often by \( \rho \) is in \( F_i \). The score of a set \( F \) of vertices measures how often \( F \) has been visited completely since the last visit of a vertex not in \( F \). McNaughton proved the existence of strategies for the winning player that bound her opponent’s scores by \(|A|!\) [1], provided the play starts in her winning region. Such a strategy is necessarily winning. The bound \(|A|!\) was subsequently improved to 2 (and shown to be tight) [2]. Thus, the winner of a Muller game can be determined by solving a (much simpler, albeit large) safety game. In the following, we present a novel algorithm and a novel type of memory structure for Muller games derived from solving this safety game. We also obtain a natural quality measure for strategies in Muller games and are able to extend the definition of permissiveness [3] from parity games to Muller games.

In the following, we use the notions of winning strategies and winning regions as defined in [4].

II. SCORING FUNCTIONS FOR MULLER GAMES

We begin by introducing scoring functions. For a more detailed treatment we refer to [2], [1].

Definition 1. Let \( w \in V^* \), \( v \in V \), and \( \emptyset \neq F \subseteq V \).

- Define \( \text{Sc}_F(z) = 0 \).
- If \( v \notin F \), then \( \text{Sc}_F(wv) = 0 \) and \( \text{Acc}_F(wv) = \emptyset \).
- If \( v \in F \) and \( \text{Acc}_F(w) = F \setminus \{v\} \), then \( \text{Sc}_F(wv) = \text{Sc}_F(w) + 1 \) and \( \text{Acc}_F(wv) = \emptyset \).
- If \( v \in F \) and \( \text{Acc}_F(w) \neq F \setminus \{v\} \), then \( \text{Sc}_F(wv) = \text{Sc}_F(w) \) and \( \text{Acc}_F(wv) = \text{Acc}_F(w) \cup \{v\} \).

Now, let \( w, w' \in V^* \) and \( F \subseteq 2^V \).

1) \( w \) is \( F \)-smaller than \( w' \), denoted by \( w \preceq_F w' \), if \( \text{Last}(w) = \text{Last}(w') \) and for all \( F' \in F \):
- \( \text{Sc}_F(w) < \text{Sc}_F(w') \), or
- \( \text{Sc}_F(w) = \text{Sc}_F(w') \) and \( \text{Acc}_F(w) \subseteq \text{Acc}_F(w') \).

2) \( w \) and \( w' \) are \( F \)-equivalent, denoted by \( w \equiv_F w' \), if \( w \preceq_F w' \) and \( w' \preceq_F w \).

Our results rely on the following lemma.

Lemma 1 [2]. In every Muller game \( G = (A, F_0, F_1) \), Player \( i \) has a winning strategy that bounds every \( \text{Sc}_F \) with \( F \in F_{1-i} \) by two during every consistent play.

Hence, a player wins the Muller game if and only if she can prevent her opponent from ever reaching a score of three. This is a safety condition!

III. SOLVING MULLER BY SOLVING SAFETY

Fix a Muller game \( G = (A, F_0, F_1) \) and consider the following safety game \( G_S \): the scores and accumulators of Player 1 are tracked up to threshold three by the arena. More formally, we take the \( =_{F_1} \)-quotient of the unraveling of \( A \) up to the positions where Player 1 reaches a score of three for the first time. Player 1 wins a play in this (finite) arena, if he reaches a score of three. Hence, Player 0 wins if her opponent never reaches a score of three.

---

*This work was supported by the projects Games for Analysis and Synthesis of Interactive Computational Systems (GASICS) and Logic for Interaction (LINT) of the European Science Foundation.
Theorem 1. Let $\mathcal{G}$ be a Muller game with vertex set $V$. One can effectively construct a safety game $\mathcal{G}_S$ with vertex set $V^S$ and a mapping $f : V \to V^S$ with the following properties:

1. For every $v \in V$: Player $i$ wins the Muller game from $v$ if and only if she wins the safety game from $f(v)$.
2. Player 0 has a finite-state winning strategy for $\mathcal{G}$ whose set of memory states is $V^S$.
3. $|V^S| \leq (|V|^3)^3$.

Note that the first statement speaks about both players while the second one only speaks about Player 0. This is due to the fact that the safety game keeps track of Player 1’s scores only. To obtain a winning strategy for Player 1, we have to track Player 0’s scores. The first claim follows directly from Lemma 1 while the second one is proved by turning the winning region of Player 0 in $\mathcal{G}_S$ (restricted to the vertices reachable via a positional winning strategy for $\mathcal{G}_S$) into a memory structure whose strategy prevents Player 1 from reaching a score of three in $\mathcal{G}$. Such a strategy is winning. The size of this memory structure is at most cubically then the size of the LAR memory structure.

Furthermore, by only using the $\leq_{X^*}$-maximal elements of Player 0’s winning region as memory states, one obtains an even smaller memory structure that still implements a winning strategy. On the other hand, by using all vertices in the winning region, but using the most general non-deterministic winning strategy for Player 0 in $\mathcal{G}_S$ (cf. [3]), we also obtain the most general non-deterministic winning strategy that prevents the losing player from reaching a score of three (which can obviously be generalized to any threshold $k$). This extends the notion of permissive strategies from parity to Muller games.

IV. SAFETY REDUCTIONS FOR INFINITE GAMES

Since Muller conditions are on a higher level of the Borel hierarchy than safety conditions, there is no game reduction from Muller to safety games (using the notion of reduction as defined, e.g., in [4]). Nonetheless, we have just solved a Muller game by solving a safety game. The price we have to pay is that we only obtain a winning strategy for one player while standard reductions yield winning strategies for both. Next, we present a general construction comprising our result.

Definition 2. A game $\mathcal{G} = (A, \text{Win})$ with vertex set $V$ and set $\text{Win} \subseteq V^\omega$ of winning plays for Player 0 is (finite-state) safety reducible, if there is a regular language $L \subseteq V^*$ of finite words such that:

- For every play $\rho \in V^\omega$: If $\text{Pref}(\rho) \subseteq L$, then $\rho \in \text{Win}$.
- If Player 0 wins from $v$, then she has a strategy $\sigma$ such that $\text{Pref}(\sigma) \subseteq L$ for every $\rho$ consistent with $\sigma$ and starting in $v$.

Note that a strategy $\sigma$ satisfying the second property is winning for Player 0 from $v$. Many solution algorithms for games can be phrased in this terminology, e.g., the progress measure algorithms for parity games [5] respectively Rabin and Streett games [6], as well as work on bounded synthesis [7] and LTL realizability [8].

Theorem 2. Let $\mathcal{G}$ be a game with vertex set $V$ that is safety reducible with language $L(\mathcal{A})$ for some DFA $\mathcal{A} = (Q, V, q_0, \delta, F)$. Define the safety game $\mathcal{G}' = (A \times \mathcal{A}, V \times F)$. Then:

1. For every $v \in V$, Player 0 wins the $\mathcal{G}$ from $v$ if and only if she wins $\mathcal{G}'$ from $(v, \delta(q_0, v))$.
2. Player 0 has a finite-state winning strategy for $\mathcal{G}$ with memory states $Q$.

This results gives a unified approach to solving parity, Rabin, Streett, and Muller games (and many more) by solving safety games. Furthermore, the notion of safety-reduction allows to generalize permissiveness to all these games, yielding what one could call $L$,-permissiveness, i.e., we obtain the most general non-deterministic winning strategy that “stays” in $L$.

V. CONCLUSION

We have shown how to translate a Muller game into a safety game to determine both winning regions and a finite-state winning strategy for one player. Then, we generalized this construction to a new type of reduction from infinite games to safety games with the same properties. The reduction from Muller to safety games is implemented in the tool GAVS+ [9].

The quality of a strategy can be measured by the maximal score value the opponent can achieve. We conjecture that there is no tradeoff between size and quality of a strategy.

Finally, there is a tight connection between permissive strategies, progress measure algorithms, and safety reductions for parity games. Whether the safety reducibility of Muller games can be turned into a progress measure algorithm is subject to ongoing research.

Acknowledgments. The authors want to thank Wolfgang Thomas for bringing McNaughton’s work to their attention, Vladmir Fridman for fruitful discussions, and Chih-Hong Cheng for his implementation of the algorithm.

REFERENCES


1 See www6.in.tum.de/~chengh/gav/ for details and to download the tool.
Interaction Graphs: Additives

Thomas Seiller
LAMA UMR 5127 CNRS — Université de Savoie
Bâtiment Chablais, Campus Scientifique
73376 Le Bourget-du-Lac Cedex, France
Email: thomas.seiller@univ-savoie.fr

Abstract—Geometry of Interaction (GoI) is a kind of semantics of linear logic proofs that aims at accounting for the dynamical aspects of cut-elimination. We present here a parametrized construction of a Geometry of Interaction for Multiplicative Additive Linear Logic (MALL) in which proofs are represented by families of directed weighted graphs. This construction is founded on a geometrical identity between sets of cycles which generalizes the three-term adjunctions which sustain usual GoI constructions. Contrarily to former Geometries of Interaction dealing with additive connective, the proofs of MALL are interpreted by finite objects in this model. Moreover, we show how one can obtain, for each value of the parameter, a denotational semantics for MALL.

I. GEOMETRY OF INTERACTION

The Geometry of Interaction program [5] was introduced by Girard a couple of years after his discovery of Linear Logic [6]. It aims at giving a semantics of linear logic proofs that would account for the dynamical aspects of cut-elimination, hence of computation through the proofs-as-program correspondence. Informally, a GoI consists of:

• a set of mathematical objects — paraproofs — that will contain, among other things, the interpretations of proofs (or λ-terms);

• a notion of execution that will represent the dynamics of cut-elimination (or β-reduction).

Then, from these basic notions, one should be able to "reconstruct" the logic from the way the paraproofs interact:

• From the notion of execution, one defines a notion of orthogonality between the paraproofs that will allow to define formulas — types — as sets of paraproofs closed under bi-orthogonality (a usual construction in realizability). The notion of orthogonality should be thought of as a way of defining negation based on its computational effect.

• The connectives on formulas are defined from "low-level" operations on the paraproofs, following the idea that the rules governing the use of a connective should be defined by the way this connective acts at the level of proofs, i.e. by its computational effect.

Throughout the years, Girard defined several such semantics, mainly based on the interpretation of a proof as an operator on an infinite-dimensional Hilbert space. It is worth noting that the first version of GoI [2] allowed Abadi, Gonthier, and Levy [7] to explain the optimal reduction of λ-calculus defined by Lamping [8].

The latest version of GoI [1], from which this work is greatly inspired, is based on the interpretation of proofs as operators in the type II₁ hyperfinite factor. It is related to quantum coherent spaces [9], which suggest future applications to quantum computing. Moreover, the great generality and flexibility of the definition of exponentials also seem promising when it comes to the study of complexity1.

II. INTERACTION GRAPHS

In a previous work [11], we developed a graph-theoretical GoI for Multiplicative Linear Logic (MLL) where proofs are interpreted by finite objects2. In this work [12] we show how these graph-theoretical constructions can be generalized to finite formal weighted sums of graphs, and how to construct additive connectives in this setting. Contrarily to what happens in the two other versions of GoI dealing with additives [4], [1], proofs of MALL are interpreted in our framework by finite objects.

This construction is based on a geometrical identity — called the cyclic property — between sets of cycles which is a generalization of GoI's three-term adjunctions. As a consequence of the cyclic property, one gets a three-term geometrical adjunction. From this geometrical adjunction and any quantifying map from the interval [0, 1] to $\mathbb{R}_{>0} \cup \{\infty\}$, one can define a parametrized notion of orthogonality and construct a geometry of interaction for MALL. We are therefore able to define not just one but a whole family of models, two of which will turn out to be combinatorial versions of Girard's older [2], [3], [4] and more recent [1] GoI constructions.

We then proceed to show how, from any of these models, one can obtain a ∗-autonomous category $\mathbb{Graph}_{\text{MALL}}$ with $\forall \not\equiv \otimes$ and $1 \not\equiv \bot$, i.e. a non-degenerate denotational semantics for MLL. As in all the versions of GoI dealing with additive connectives, our construction of additives does not define a categorical product. However, it is possible to define an internal notion of observational equivalence within the model. As a consequence of the cyclic property, we can show that this notion of observational equivalence is a

1 A first result in this direction was obtained by Girard [10] who obtained a characterization of the class NL.

2 Even though the graphs we consider can have an infinite set of edges, linear logic proofs are represented by finite graphs.
congruence on the category $\mathcal{G}raph_{\text{MLL}}$. We are then able to define a categorical product from our additive connectives when considering classes of observationally equivalent objects, obtaining a denotational semantics for MALL with additive units. This category is non-degenerate in the sense that no connectives and units are equal and neither the mix rule nor the weakening hold. This model is moreover obtained as a full subcategory of a $*$-autonomous category, i.e. a model of MLL with units.

Finally, we show how our framework is related to Girard’s versions of GoI. A first choice of quantifying map gives us a model that can be embedded in Girard’s GoI5 framework. It can be shown that it is a combinatorial version of (the multiplicative additive) fragment of GoI5. In particular, the orthogonality defined in our setting corresponds to Girard’s notion of orthogonality based on the Fuglede-Kadison determinant\(^3\). Our construction therefore gives insights on the notion of orthogonality used by Girard, and his use of idioms and pseudo-trace which both have a graph-theoretical counterpart in our construction.

We then proceed to show that a second choice of quantifying map defines a model where orthogonality is defined by nilpotency: our construction thus defines in this case a (refined) version of older Geometries of Interaction [2], [3], [4]. However, this choice of parameter yields a trivial model of the additives. Nonetheless, the interaction graphs make a bridge between "old-style" geometry of interaction and Girard’s most recent work [1], unveiling in the process the unique geometrical identity — the so-called cyclic property — behind those seemingly different constructions.

---

\(^3\)The Fuglede-Kadison determinant is a generalization of the determinant defined in any type II\(_1\) factor [13].

---

**REFERENCES**

Ribbon Proofs for Separation Logic

John Wickerson  
University of Cambridge  
john.wickerson@cl.cam.ac.uk

Mike Dodds  
University of Cambridge  
mike.dodds@cl.cam.ac.uk

Matthew Parkinson  
Microsoft Research Cambridge  
mattpark@microsoft.com

A program proof should not merely certify that a program is correct; it should explain why it is correct. A proof should be more than ‘true’: it should be informative, and it should be intelligible. Extending work by Bean [1], we introduce a system that produces readable program proofs that are highly scalable and easily modified.

The de facto standard for presenting program proofs in Hoare logic [2] is the \textit{proof outline}, in which the program’s instructions are interspersed with ‘enough’ assertions to allow the reader to reconstruct the derivation tree. As an example, Fig. 1a presents a proof outline for a program that performs in-place list reversal. A key asset of the proof outline is what we shall call \textit{instruction locality}: that one can verify each instruction in isolation (by confirming that the assertions immediately above and below it form a valid Hoare triple) and immediately deduce that the entire proof is correct.

The proof outline suffers several drawbacks, however. First, there is much repetition: ‘\texttt{list \alpha x}’ appears redundantly in six consecutive assertions before it is used on line 25. Second, there is no distinction between those parts of an assertion that are affected by an instruction and those that are merely in what separation logic calls the \textit{frame}. For instance, line 19 affects only the second and fourth conjuncts of its preceding assertion, but it is difficult to deduce its effect because two unchanged conjuncts are interspersed. (Had we followed common practice and reduced the size of the proof outline by combining this line with the assignment on line 17, the effect would be even harder to deduce.) Third, the use of logical variables is unclear. For instance, spotting that the $\beta$ in line 20 differs from the one in line 22 requires careful examination, or else, as we have done, an explicit textual comment. These minor problems in our illustrative example quickly become devastating when scaled to large programs.

Separation logic [3], [4] provides a mechanism for handling a second dimension of locality: \textit{resource locality}. One can use separation logic’s \textit{small axioms} to reason about an instruction operating only on the resources (i.e. memory cells) that it needs, and immediately deduce its effect on the entire state using the \textit{frame rule}. To depict this mechanism in a proof outline, one must show applications of the frame rule explicitly. But this is tedious; moreover, it is difficult to know when and what to frame. Meanwhile, the ribbon proof inherently supports resource locality. Its primitive steps correspond exactly to the small axioms. It is thus an ideal representation for exploiting both forms of locality that separation logic provides.

Figure 1b recasts our proof as a ribbon proof. The state is distributed across several \textit{ribbons} (thick borders). Horizontally separated ribbons describe disjoint parts of the state. The instructions are in grey bars, and the scope of each logical variable is delimited by an \textit{existential box} (thin borders). We are free to stretch ribbons as required by the layout, and, because $*$ is commutative, we can ‘twist’ them too. A temporarily inactive ribbon slides discreetly down the side of the proof. This effect is achieved by invoking the frame rule at each instruction; but crucially in a ribbon proof, these invocations are implicit and do not increase the diagram’s complexity. Observe that the repetition has disappeared, and that each instruction’s effect is clear: it affects exactly those assertions directly above and below it, while framed assertions (which must not mention variables written by the instruction) bypass to the left or right. Existential boxes extend vertically to indicate the range of steps over which the same witness is used, thus making the usage of logical variables visually clear.

In our full paper [5]:

- we present an Isabelle-checked graph-based formalisation of our proof system;
- we showcase, with a ribbon proof of the memory manager from Version 7 Unix, the ability of our diagrams to present readable proofs of large, complex programs; and
- we describe a prototype tool for mechanically checking ribbon proofs in Isabelle. Provided with a small proof script for each primitive step, our tool assembles a script that verifies the entire diagram. The tool handles tediums such as the associativity and commutativity of $*$ automatically, leaving the user to concentrate on the interesting parts of the proof.

This work lays the foundations for a new way to use logic to understand programs. Where a proof outline essentially flattens a proof to a list of assertions, our system produces geometric objects that illuminate the structure of proofs, and which can be navigated, modified and simplified by leveraging human visual intuition.

References

Fig. 1. Two proofs of list reverse. For a binary relation r, we write \( r \upharpoonright y \) for \( r \cap \mathcal{Y} \cap \mathcal{E} \). We write \( \cdot \) for sequence concatenation, \( (-)^\uparrow \) for sequence reversal and \( \varepsilon \) for the empty sequence, and define \( \mathcal{L} \) as the smallest predicate satisfying \( \mathcal{L} \cdot \mathcal{Y} = 0 + \mathcal{L} \cdot \varepsilon = \mathcal{L} \cdot \mathcal{X} = 0 \).
Short Papers from the Specific Short Papers Call
Logical Bell Inequalities

Samson Abramsky
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, U.K.
samson@comlab.ox.ac.uk

Lucien Hardy
Perimeter Institute, 31 Caroline Street North
Waterloo, ON N2L 2Y5, Canada
lhardy@perimeterinstitute.ca

Bell inequalities [3], [4] play a fundamental rôle in quantum information and computation and the foundations of quantum mechanics. They provide the basis for experimental demonstrations of non-locality and entanglement as fundamental non-classical resources for information processing. There is a huge literature on Bell inequalities, with many ingenious derivations of families of inequalities. However, a unifying principle with a clear conceptual basis has proved elusive. In [2], we introduce a form of Bell inequality based on logical consistency conditions, which we call logical Bell inequalities. This approach is both conceptually illuminating and technically powerful. We show that a rational inequality is satisfied by all non-contextual models if and only if it is equivalent to a logical Bell inequality. Thus quantitative tests for contextuality or non-locality always hinge on purely logical consistency conditions. We obtain explicit descriptions of complete sets of inequalities for the convex polytope of non-contextual probability models, and the derived polytope of expectation values for these models. Moreover, these results are obtained at a high level of generality; they apply not only to the familiar cases of Bell-type scenarios, for any number of parties, but to all Kochen-Specker configurations, and in fact to any family of sets of compatible measurements. This generality is achieved by working with measurement covers, following the sheaf-theoretic approach to non-locality and contextuality introduced by the first author and Adam Brandenburger in [1].

We also obtain results for a number of special cases. We show that a model achieves maximal violation of a logical Bell inequality if and only if it is strongly (or maximally) contextual. We show that all Kochen-Specker configurations lead to maximal violations of logical Bell inequalities in a state-independent fashion. We also derive specific violations of logical Bell inequalities for models which are possibilistically contextual, meaning that they admit logical proofs of contextuality. Well-known examples of such models are those arising from the construction in [6].

Inspiration for the present work was drawn from [5], which derives some particular cases of logical Bell inequalities. Developing these ideas in the general setting provided by [1] proves to be fruitful, and indicates the potential for a structural approach to quantum foundations, using tools developed in the semantics of computation.

We believe that the logical perspective opened up on Bell inequalities in this work, in a very general setting, will be accessible and interesting to computer scientists working in logical foundations of information and computation.

We shall illustrate the ideas by showing how the essential point can be conveyed in a very simple setting, without presupposing quantum mechanics or any other specific theory.

A. A Simple Observation

We begin with a simple and very general scenario. Suppose we have propositional formulas \( \varphi_1, \ldots, \varphi_N \). We suppose further that we can assign a probability \( p_i \) to each \( \varphi_i \).

In particular, we have in the mind the situation where the boolean variables appearing in \( \varphi_i \) correspond to empirically testable quantities; \( \varphi_i \) then expresses a condition on the outcomes of an experiment involving these quantities. The probabilities \( p_i \) are obtained from the statistics of these experiments.

Now let \( P \) be the probability of \( \Phi := \bigwedge_i \varphi_i \). Using elementary probability theory, we can calculate:

\[
1 - P = \text{Prob}(\neg \Phi) = \text{Prob}(\bigvee_i \neg \varphi_i) \leq \sum_i \text{Prob}(\neg \varphi_i) = \sum_i (1 - p_i) = N - \sum_i p_i.
\]

Tidying this up yields \( \sum_i p_i \leq N - 1 + P \).

Now suppose that the formulas \( \varphi_i \) are jointly contradictory; i.e. \( \Phi \) is unsatisfiable. This implies that \( P = 0 \). Hence we obtain the inequality

\[
\sum_i p_i \leq N - 1.
\]

This is an example of a logical Bell inequality. This basic form is generalized in [2] using a notion of \( K \)-consistency, related to the well-known MAX-SAT problem in computational complexity [7].

B. A Curious Observation

Quantum Mechanics tells us that we can find propositions \( \varphi_i \) describing outcomes of certain measurements, which not only can but have been performed. From the observed statistics of these experiments, we have very highly confirmed probabilities \( p_i \). These propositions are easily seen to be jointly contradictory. Nevertheless, the inequality

\[
\sum_i p_i \leq N - 1
\]

is satisfied by all non-contextual models if and only if it is strongly (or maximally) contextual. This is an example of a logical Bell inequality, which is basic form is generalized in [2] using a notion of \( K \)-consistency, related to the well-known MAX-SAT problem in computational complexity [7].
is observed to be strongly violated. In fact, the maximum violation of 1 can be achieved.

How can this be?

We shall discuss how this puzzle can be resolved in the presentation, and explain the striking conceptual consequences which follow.

C. Probabilistic models of experiments

We consider a standard setting for discussions of non-locality, where a number of agents each has the choice of one of several measurement settings; and each measurement has a number of distinct outcomes.

For example, consider the following table.

<table>
<thead>
<tr>
<th></th>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(0, 1)</th>
<th>(1, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, b)</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>(a, b')</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td>3/8</td>
</tr>
<tr>
<td>(a', b)</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td>3/8</td>
</tr>
<tr>
<td>(a', b')</td>
<td>1/8</td>
<td>3/8</td>
<td>3/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

Here we have two agents, Alice and Bob. Alice can choose from the settings $a$ or $a'$, and Bob can choose from $b$ or $b'$. These choices correspond to the rows of the table. The columns correspond to the joint outcomes for a given choice of settings by Alice and Bob, the two possible outcomes for each individual measurement being represented by 0 and 1. The numbers along each row specify a probability distribution on these joint outcomes.

A standard version of Bell’s theorem uses the probability table given above. This table can be realized in quantum mechanics, e.g. by a Bell state, written in the $Z$ basis as

$$\frac{|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle}{\sqrt{2}},$$

subjected to spin measurements in the $XY$-plane of the Bloch sphere, at a relative angle of $\pi/3$.

Logical analysis of the Bell table: We now pick out a subset of the elements of each row of the table, as indicated in the following table.

<table>
<thead>
<tr>
<th></th>
<th>(0, 0)</th>
<th>(1, 0)</th>
<th>(0, 1)</th>
<th>(1, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, b)</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>(a, b')</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td>3/8</td>
</tr>
<tr>
<td>(a', b)</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
<td>3/8</td>
</tr>
<tr>
<td>(a', b')</td>
<td>1/8</td>
<td>3/8</td>
<td>3/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

If we read 0 as true and 1 as false, the highlighted positions in the table are represented by the following propositions:

$$\varphi_1 = a \land b \lor \neg a \land \neg b = a \leftrightarrow b$$
$$\varphi_2 = a \land b' \lor \neg a \land \neg b' = a \leftrightarrow b'$$
$$\varphi_3 = a' \land b \lor \neg a' \land \neg b = a' \leftrightarrow b$$
$$\varphi_4 = \neg a' \land b' \lor a' \land \neg b' = a' \oplus b'.$$

These propositions are easily seen to be contradictory. Indeed, starting with $\varphi_4$, we can replace $a'$ with $b$ using $\varphi_3$, $b$ with $a$ using $\varphi_1$, and $a$ with $b'$ using $\varphi_2$, to obtain $b' \oplus b'$, which is obviously unsatisfiable.

We see from the table that $p_i = 1$, $p_i = 6/8$ for $i = 2, 3, 4$. Hence the violation of the Bell inequality is 1/4.

We shall discuss other examples in the presentation, emphasizing those in which purely logical proofs of non-locality can be given. Details can be found in [2].

REFERENCES

Approximations of Conjunctive Queries

Pablo Barceló  
Department of Computer Science  
University of Chile  
Santiago, Chile  
pbarcelo@dcc.uchile.cl

Leonid Libkin  
School of Informatics  
University of Edinburgh  
Edinburgh, UK  
libkin@inf.ed.ac.uk

Miguel Romero  
Department of Computer Science  
University of Chile  
Santiago, Chile  
mromero@ing.uchile.cl

Abstract—The term conjunctive queries refers to the \( \exists, \land \)-fragment of first-order predicate logic. These are the most common queries used in relational database systems as they capture the operations of selection, projection, and join of relational algebra. While data complexity of these queries is very low, combined complexity is well-known to be \( \mathsf{NP} \)-complete. Given the importance of conjunctive queries, subclasses guaranteeing tractable combined complexity have been extensively studied; those include classes of acyclic queries, as well as queries of bounded treewidth and bounded hypertree width.

We have recently initiated a study of approximations of conjunctive queries from the static analysis point of view. The key idea is to approximate “bad” (intractable) conjunctive queries by ones from “good” (tractable) classes. The paper appears at PODS’2012 and combines logical techniques with those from graph theory, especially theory of graph homomorphisms. We believe a brief account of this work could be of interest to the LICS community. Specifically, in PODS’12 paper we showed that approximations exist for those include classes of acyclic queries, as well as queries of bounded treewidth and bounded hypertree width.

I. INTRODUCTION

The concept of approximating queries by those that are easier to execute is very common in database research [10]. We have recently initiated a study of approximations of conjunctive queries [2]. These queries, which form the \( \exists, \land \)-fragment of first-order logic, play a special role in database applications [1]. In addition, we have a very good understanding of their complexity. While in general their combined complexity is \( \mathsf{NP} \)-complete [3], tractable classes of conjunctive queries have been identified; these include acyclic queries, queries of bounded treewidth, and queries of bounded hypertree width, see [5], [6], [7], [9], [11]. We studied approximations from the point of view of static analysis, i.e., we looked for approximations independent of the input database: for a query \( Q \), we want to find another query \( Q' \) that will be much faster than \( Q \), and whose output would be close to the output of \( Q \) on all databases.

We have shown in [2] that approximations always exist for classes mentioned above; furthermore, sizes of approximations are at most polynomial in the size of the original query (and often they do not exceed the size of the original query), and they can be found in at most single-exponential time.

To see why these properties are desirable, consider, for instance, the complexity of checking whether a tuple \( \bar{a} \) belongs to the output of a conjunctive query \( Q \) on a database \( D \). This is of the order \( |D|^{O(|Q|)} \), where \( |\cdot| \) measures the size of a database or a query [1]; the problem is actually known to be \( \mathsf{NP} \)-complete [3]. The data complexity of conjunctive query evaluation is of course very low (in \( \mathsf{AC}^0 \)), but if we are concerned with evaluating queries over large data sets, having \( O(|Q|) \) as the exponent may be too high. With conjunctive queries from the good classes mentioned above, the \( O(|Q|) \) exponent is replaced by a fixed one. For instance, for acyclic conjunctive queries, evaluation can be done in time \( O(|D| \cdot |Q|) \) [11]. Thus, assuming that we can find an approximation relatively fast (for instance, in time \( 2^{O(|Q| \log |Q|)} \)) and its size is roughly bounded by the size of \( Q \), we may want to check if the approximation gives us the answer first. The complexity of doing so is \( 2^{O(|Q| \log |Q|)} + O(|D| \cdot |Q|) \), which is likely to be much less than \( |D|^{O(|Q|)} \) on very large databases \( D \).

II. DEFINITION OF APPROXIMATIONS

Suppose we have a query, i.e., a \( (\exists, \land) \)-FO formula \( Q(x) \), and we want to approximate it by \( Q'(x) \) coming from a good class of queries \( C \). There are two requirements for this:

1) \( Q' \) should not return false results, i.e., \( Q' \subseteq Q \) (that is, \( Q'(D) \subseteq Q(D) \) for all databases \( D \));

2) we cannot find a better approximating query, i.e., there is no \( Q'' \in C \) satisfying \( Q'' \subseteq Q' \subseteq Q \).

Such a query \( Q' \), if it exists, is called a \( C \)-approximation of \( Q \).

The classes we consider are divided into graph-based and hypergraph-based classes. Both are obtained from the syntax of the query. Given a query \( Q \), we define its graph \( G(Q) \) and its hypergraph \( H(Q) \) as graphs of hypergraphs whose nodes are variables in \( Q \). In \( G(Q) \), there is an edge between \( x \) and \( y \) if they occur in the same atomic subformula. In \( H(Q) \), each atomic subformula over variables \( x_1, \ldots, x_m \) generates a hyperedge \( \{x_1, \ldots, x_m\} \). Now the classes are:

- Hypergraph-based classes: these are classes \( \mathsf{HTW}(k) \) of queries \( Q \) such that hypertree width (see [6]) of \( H(Q) \) is at most \( k \). The class \( \mathsf{HTW}(1) \) is also known as the class of acyclic queries.

- Graph-based classes: these are classes \( \mathsf{TW}(k) \) of queries \( Q \) such that the treewidth of \( G(Q) \) is at most \( k \). As already mentioned, these are the standard tractable classes of conjunctive queries. Graph- and hypergraph-based notions are in general incompatible [4].
The notion of approximations can be reformulated in terms of homomorphisms of graphs (and relational structures), connecting it with many concepts studied in the field of constraint satisfaction. Since $Q \subseteq Q'$ iff there is a homomorphism $T_Q' \rightarrow T_Q$ from the tableau $T_{Q'}$ of $Q'$ to the tableau $T_Q$ of $Q$, finding an approximation in class $C$ amounts to finding the largest elements of $C$ below a given structure in the lattice of structures and homomorphisms [8]. In fact we use techniques of [8] extensively in our proofs.

III. SUMMARY OF RESULTS

A. Graph-based notions

Let $C$ be a class of graphs closed under taking subgraphs. Assume that a single-element loop is in $C$ (it has treewidth 1 and is harmless). Slightly abusing notation, we also use $C$ to denote the class of queries $Q$ with $G(Q) \in C$.

We show the following:

1) every conjunctive query has a $C$-approximation;
2) each approximation is equivalent to one that has at most as many atomic subformulas as $Q$ (i.e., at most as many subgraphs as $Q$);
3) an approximation can be constructed in time $2^{O(|Q| \log |Q|)}$;
4) there are at most exponentially many non-equivalent approximations of $Q$ (and the exponential number of approximations can be witnessed).

We also study the complexity of the problem of recognizing approximations: given $Q$ and $Q'$, check whether $Q'$ is a $C$-approximation of $Q$. We show that when $C = \text{TW}(k)$, for $k \geq 1$, this problem is DP-complete. This holds even if $Q, Q'$ are minimized, i.e., their tableaux are cores.

We then look what happens when we ask queries over graphs, i.e., the vocabulary of a single binary relation $E(\cdot, \cdot)$. We show (for Boolean queries), the following trichotomy. If $G(Q)$ is not bipartite, then only a trivial approximation $\exists x \ E(x, x)$ exists. If $G(Q)$ is bipartite and has an unbalanced directed cycle (with a different number of forward and backward edges), then the only approximation is $\exists x, y \ E(x, y) \wedge E(y, x)$ (which is essentially trivial since every query with a bipartite graph contains this query). But if $G(Q)$ is bipartite and has no unbalanced cycles, then $Q$ has nontrivial approximations with strictly fewer subgoals than $Q$ itself.

While the condition above may seem restrictive, it only applies to queries over graphs; with higher arities it is easier to get nontrivial approximation (we show an example later).

B. Hypergraph-based notions

For hypergraph-based classes, we need a more complex closure condition for the existence of approximations if we want to capture classes such as $\text{HTW}(k)$. Specifically, we require that hypergraphs be closed under two operations:

- restriction to induced subhypergraphs;
- extensions of hyperedges with fresh nodes (not used in other hyperedges).

We prove that each class $HTW(k)$ satisfies these conditions. Then, we show that exactly the same existence results as those mentioned in the graph-based subsection hold. Hence, we get approximations for all the classes of conjunctive queries that admit tractable evaluation.

IV. A COUPLE OF EXAMPLES

We start with the simplest case. Boolean queries over graphs, and write then in the standard rule-based notation for conjunctive queries. Consider a query $Q_1: \exists x, y, z \ E(x, y), E(y, z), E(z, x)$. Its graph is not bipartite and its best acyclic approximation is trivial: $Q'_1: \exists x, y, z \ E(x, y), E(y, z), E(z, x)$. While the condition above may seem restrictive, it only applies to queries over graphs, i.e., the vocabulary of a single binary relation $E(\cdot, \cdot)$. When $Q_2$ is bipartite and has no unbalanced cycles, then the only approximation is trivial. If we had a binary relation instead and omitted the middle variable, then we get a 3-cycle. But it has nontrivial acyclic approximations.

With higher arities, we get richer approximations within both graph-based and hypergraph-based classes. Consider, for instance, a query over ternary relations: $Q_3: \exists x, y, z, u, v \ E(x, y, z), E(y, z, u), E(z, u, v)$. It is a slight extension of the bad query $Q_1$ (just remove syntactically $x$ and you get a 3-cycle). But it has nontrivial $\text{TW}(1)$-approximations (the best for graph-based classes), e.g., $Q'_3: \exists x, y, z \ E(x, y, z)$, $E(y, x, u)$, $E(y, u, v)$.

As another example, consider a query

$Q_4: \exists x, y, z \ E(x, y, z), E(z, u, v), E(u, v, x)$.

If we had a binary relation instead and omitted the middle attribute, we again would obtain a query whose tableau is a 3-cycle, thus having only trivial approximations. However, going beyond graphs lets us find nontrivial hypergraph-based (particularly, acyclic) approximations. In fact this query has several non-equivalent acyclic approximations, for instance $Q'_4: \exists x, y, z \ E(x, y, z), E(z, u, v)$, $E(u, v, x)$.

REFERENCES

Abstract—We present a general model-theoretic technique that we developed and used in [3], [4] to obtain complete axiomatizations of fragments of MSO on finite trees. There is much interest in studying logics on finite trees, and many logics of interest are fragments of MSO. Previously FO axiomatizations were known. To produce axiomatizations beyond FO, we had to develop a new technique that combines classical tools from infinite model theory (Henkin semantics for higher-order logics) with those more typical in finite model theory (Ehrenfeucht-Fraissé games, and their composition). The key idea behind the technique is to analyze infinite Henkin models of our axioms, and use games to show that they are elementarily equivalent to finite trees.

Given the general interest in the LICS community in logics on finite trees, and a new set of tools developed by us (that combine classical and finite model theory), we believe that a brief account of this work will be of interest to the LICS audience.

I. INTRODUCTION

Recently there has been much interest in studying logics over ordered unranked trees, mainly due to connections with XML research, since labeled unranked trees serve as a standard abstraction of XML documents. Logics are used to describe the structure of XML documents, and to query data they contain, and absolute majority of those used in this context happen to be fragments of MSO (see [8]).

The goal of this work is to obtain complete axiomatizations of MSO and its fragments on finite node-labeled sibling-ordered trees. Such axiomatizations have previously been presented for FO-theories [5] but extending the work to MSO presents a number of challenges. We address those by developing a new model-theoretic technique by which we obtain complete axiomatizations not only of MSO but also of some of its fragments, such as monadic transitive closure logic and monadic least fixed-point logic.

The new tools we developed combine traditional model-theoretic techniques used to show completeness, with techniques more common in finite model theory, namely Ehrenfeucht-Fraissé games as well as techniques for composing games. These results have not been presented in the forums traditionally attended by LICS attendees: the conference version appeared in [3] (the journal version is to appear in [4]). As this work addresses traditional LICS topics via using a new set of techniques, we believe that the LICS community could be interested in a short presentation of this work.

To give a flavor of our results and techniques, below we describe the approach for MSO. We give a brief account of the key ingredients: Henkin completeness and Feferman-Vaught theorems, that we need to obtain our results. References [3], [4] can be consulted for details, as well as for extensions to various MSO fragments.

II. HENKIN COMPLETENESS

It is well known that MSO is highly undecidable on arbitrary standard structures and hence not recursively enumerable. However, Henkin [7] formulated a non-standard semantics for logics of even higher order, and showed that under this interpretation, they can be completely axiomatized. In the case of MSO, the procedure amounts to allowing “non-standard” or “Henkin” interpretations of MSO-formulas in addition to their standard interpretations. In such non standard structures, the set quantifier is interpreted as ranging not over the whole powerset of the domain, but over one of its explicitly given subsets, required to satisfy some good closure conditions. This means that each Henkin structure is given as a pair, containing a usual relational structure together with a subset of the powerset of its domain. A point of particular interest to us is that on finite structures, the mandatory closure conditions are only satisfied by the whole powerset of the domain. It follows that finite Henkin structures are always equivalent to standard structures. This point matters here for the following reason. As a first step of our proof, we show that our axioms are complete on the class of their Henkin models, but the problematic thing at this stage is that some of these models might not be finite trees. However, it is straightforward to infer from our axioms that a finite structure satisfies them if and only if it is a tree. Hence, if there are Henkin models of our axioms which are not finite trees, they have to be infinite. In what remains, we need to show that such infinite models can be “dismissed”.

III. FEFERMAN-VAUGHT THEOREMS

Even though our main completeness result concerns finite trees, inside the proof we need to consider infinite Henkin structures. In this context even such basic notions as substructures, as well as methods for forming new structures out of existing ones have to be redefined carefully. There
is a whole range of model-theoretic methods to form new structures out of existing ones [6], [9]. Familiar constructions like disjoint unions are redefined as particular cases of a notion of generalized product of FO-structures and abstract properties of such products are studied. Results telling how theories of complex structures can be obtained from theories of the components they are built from are known as Feferman-Vaught theorems (who proved the first such result in [6]).

Here we are particularly interested in a type of Feferman-Vaught theorem which establishes that generalized products of relational structures preserve elementary equivalence. We show such a result for a particular case of generalized product of Henkin-structures called fusion. These preservation results are shown with the crucial help of Ehrenfeucht-Fraïssé games that are suitable to use on Henkin structures. More precisely, by combining winning strategies in these games, we show that for every \( n \in \mathbb{N} \), the MSO \( n \)-theory of the fusion structure reduces to the MSO \( n \)-theories of the components structures (by MSO \( n \)-theory, we mean the restriction of the theory to MSO-formulas of quantifier depth \( n \)). We believe that such general combination techniques for Henkin structures have independent interest. We refer to [9] for an extensive discussion of the question in the more restricted context of standard structures.

IV. “REAL” COMPLETENESS

With the key ingredients – Henkin models and Feferman-Vaught theorems - we can obtain complete axiomatizations for finite trees. We cannot present the entire axiomatization in this very short abstract. The axioms are roughly subdivided into three groups: “generic” axioms true in well-behaved logics (e.g., propositional tautologies, properties of substitution), axioms stating properties of binary predicates defining the trees (e.g., transitivity of the descendant relation), and, crucially, the induction scheme.

To use the previous ingredients to obtain completeness, we define quasi-trees as Henkin models of our axioms. We then use our Feferman-Vaught results to show that MSO cannot distinguish quasi-trees from finite trees. Since finiteness is definable on trees in MSO, it then follows that every model of our axioms is a finite tree. Note that this is in sharp contrast with the FO-theory of finite trees, which does have infinite models.

Let us now briefly sketch the details of our final completeness argument. In order to proceed inductively, it is more convenient to consider a stronger version of the result concerning Henkin substructures of quasi-trees that we call quasi-forests. To grab some intuition consider a finite tree and remove the root node; then it is no longer a finite tree. Instead it is a finite sequence of trees, whose roots stand in a linear sibling order. It does not have a unique root, but it does have a unique left-most root: it is a finite forest. Now given a node \( \alpha \) in a quasi-tree \( T \), we let \( T_\alpha \) be the Henkin substructure of \( T \) generated by the set of its siblings to the right and of their descendants. We call \( T_\alpha \) a quasi-forest. Using our game composition results, we finally complete our proof by showing that for each \( n \) and for each node \( \alpha \) in a quasi-tree, the quasi-forest \( T_\alpha \) is \( n \)-equivalent to a finite forest. The argument essentially relies on an inductive axiom scheme, which for simplicity we only give here for the restricted case of MSO on finite words (\( \varphi(x) \) standing here for any definable MSO property of finite words):

\[
\forall x (\forall y ((x < y \rightarrow \varphi(y)) \rightarrow \varphi(x)) \rightarrow \forall z \varphi(z)
\]

Extending our approach to other classes of finite structures would involve finding comparable induction schemes. This suggests that other natural candidates would be fragments of MSO on classes of finite structures for which MSO satisfiability is decidable (e.g., structures of bounded treewidth).

V. CONCLUSION

In [3], [4] we obtained complete axiomatizations not only for MSO but also for monadic transitive closure and least fixed point logics on finite trees. The method we developed is quite uniform and can be used for other logics as well. While it follows the route used in modal logic, where “canonical models” are often transformed in order to obtain intended models [1], its key new element is the use of Henkin semantics: the model we first create is a Henkin model, and then we modify it to obtain a model that is among our intended ones. There are related complete axiomatizations on infinite models [2], [10], [11], where completeness proofs are based on automata-theoretic techniques, which are probably harder to adapt to obtain axiomatizations in the finite case. This leads to an intriguing question whether some of our model-theoretic techniques could also be used as an alternative to automata, in order to show other sorts of results, not necessarily related to complete axiomatizations.

REFERENCES

Uniform Polytime Computable Operators on Univariate Real Analytic Functions

Akitoshi Kawamura\(^1\) Norbert Müller Carsten Rösnick and Martin Ziegler

University of Tokyo, Japan Trier University, Germany Darmstadt University of Technology, Germany

Abstract—Kawamura and Cook (2010) have recently extended the established uniform computational complexity theory from continuous real functions to operators, based on the notion of second-order polytime computability and representations. Building on this framework we introduce two different such representations for the space of real functions analytic on (some open neighbourhood of) \([0, 1]\); and prove them a) polytime equivalent and b) reasonable in rendering the primitive operations of evaluation, addition, multiplication, anti-differentiation, and maximization polytime computable. This turns several known nonuniform results into explicit, uniform, and efficient algorithms.

I. INTRODUCTION
A. Some Previous, Nonuniform Complexity Results

Continuous functions \(f : [0, 1] \to \mathbb{R}\) admit a notion of complexity by requiring the evaluation \(x \mapsto f(x)\) to be computable in the sense of approximations up to absolute error \(\leq 2^{-n}\) within time bounded in terms of \(n\) only.

Operators have thus been investigated nonuniformly in the sense of what complexity class they map polytime computable functions to. For instance [5, 3] prove

**Fact 1.**

i) \(\max(f) = \{[0, 1] \ni x \mapsto \max_{y \leq x} f(y)\} \in C[0,1]\) is polytime computable for every polytime-comp. \(f \in C[0,1]\) iff \(\mathcal{P} = \mathcal{NP}\); cf. [4, Thm. 3.7].

ii) \(\int f = \{[0,1] \ni x \mapsto \int_0^x f(y) \, dy\} \in C[0,1]\) is polytime computable for every polytime-computable \(f \in C[0,1]\) iff \(\mathcal{P} = \#\mathcal{P}\); cf. [4, Thm. 5.33].

A real function \(f : [-1, +1] \to \mathbb{R}\) is called analytic if it is the restriction of a complex differentiable \(g : \mathbb{B}(0, 1 + \epsilon) \to \mathbb{C}\) where \(\mathbb{B}(w, r) = \{z \in \mathbb{C} : |z - w| < r\}\). By restricting the domains of the aforementioned operators to analytic functions \(f \in C^{\infty}[0,1]\), Fact 1 improves drastically [4, p.208]:

**Fact 2.** Let \(f : [0,1] \to \mathbb{R}\) be polytime computable and analytic on some complex open neighbourhood of \([0, 1]\). Then \(\max(f)\) and \(\int f\) are again (analytic and) polytime computable [7], [18].

These facts are however nonuniform in the sense that they do not describe how to obtain \(\max(f)\) and \(\int f\) from (which information about) \(f\); and so are their proofs. Thus:

**Remark 3.** The polytime complexity of \(\max(f)\) and \(\int f\) claimed in Fact 2 refers only to the dependence on the single output precision parameter \(n\): in the upper running time bound \(\leq c \cdot n^k\) both ‘constants’ \(c\) and the exponent \(k\) may and will depend on (unspecified parameters of) \(f\) in an unspecified way.

The present work aims at explicating and clarifying this dependence (Thm. 6.8), i.e., to identify (upper bounds on) the complexity of the ‘basic’ operations on real analytic functions.

B. Computing Real Functions and Operators

Passing a mathematical object to a computing device always requires some form of encoding. For objects like natural numbers, reals, real functions (or even, say, compact sets), Weihrauch’s Type-2 Theory of Effectivity (TTE) provides a fruitful framework for defining them (in form of ‘representations’, which are surjective partial mappings from Cantor space \([0, 1]^\mathbb{N}\) to the universe under consideration).

For that matter, denote by \(\rho_d\) the so-called ‘signed-digit’ representation [10, §7.2] for real numbers, and by \(\rho_\mathbb{R}\) an encoding (polytime equivalent to \(\rho_d\); [10, Lem. 7.3.5]) by a sequence of integers \([b_n]_n\) with \(|\text{real} - b_n/2^n| \leq 2^{-n}\) [4, §2]; \(\mathbb{D}_n := \{b/2^n : b \in \mathbb{Z}\}\), \(\mathbb{D} := \bigcup_n \mathbb{D}_n\).

Similarly, for investigating the computability of an operator on \(C[0,1]\), an a) encoding of continuous \(f : [0,1] \to \mathbb{R}\) has to be fixed. Moreover, to allow a discussing of ‘complexity’ of real operators, a b) notion of ‘size’ of the argument (which now is from \(C[0,1]\)) has to be defined.

[4, Def. 2.37] solves a) by representing \(f\) not as a binary string but as an oracle which, when queried \(q \in \mathbb{D}_n\), answers some \(p \in \mathbb{D}_n\) with \(|f(q) - p| \leq 2^{-n}\).

**C. Second-Order Polynomials and Representations**

[2] resolves I-B.a) by extending TTE: a real function \(f\), say, is encoded not necessarily as a sequential binary string (i.e. a mapping \(\mathbb{N} \to \{0, 1\}\)) but as a total mapping \(\varphi : \{0, 1\}^* \to \{0, 1\}^*\). More precisely \(\varphi\) is required to be length-monotone: \(|\tilde{u}| \leq |\tilde{v}| \Rightarrow |\varphi(\tilde{u})| \leq |\varphi(\tilde{v})|\). The mapping \(\ell := |\varphi| : \mathbb{N} \ni |\tilde{u}| \mapsto |\varphi(\tilde{u})| \in \mathbb{N}\) is thus well-defined — and serves as a notion of ‘size’ of the (encoded) \(f\) (which resolves I-B.b)). More generally, for \(\mathbb{R}\) denoting the set of all such (total, monotone) \(\varphi\), a second-order representation of a universe \(\mathcal{U}\) is a surjective partial map \(\delta : \subseteq \mathbb{R} \to \mathcal{U}\).

**Remark 4.** a) A length-monotone \(\varphi : \{0, 1\}^* \to \{0, 1\}^*\) is a \(\rho_\mathbb{R}\)-name of \(f \in C[0,1]\) if \(|\tilde{u}| \mapsto |\varphi(1^{\tilde{u}})|\) is a modulus of continuity of \(f\) (and subject to appropriate padding) the following holds:

\[
|\text{bin}(\varphi(0^{\tilde{u}}))/2^{\ell(u)} - f(\text{bin}(\tilde{u}))/2^{\ell(\tilde{u})})| \leq 2^{-|\tilde{u}|}. \tag{1}
\]
b) An ordinary representation $\delta \subseteq \{0,1\}^\omega \rightarrow \mathbb{U}$ induces the second-order representation $\tilde{\delta}$ as follows: $\varphi : \{0,1\}^\omega \rightarrow \{0,1\}$ is a $\tilde{\delta}$-name of $u \in \mathbb{U}$ whenever the infinite binary string $\bar{a}$ with $\sigma_{\text{binary}}(\bar{a}) = \varphi(\bar{a})$ is a $\delta$-name of said $u$.

Note that for polynomial operators to be closed under composition, a running time (for some operator) like $P(n, \ell) = \ell(\ell(n^2) \cdot n)$ should also be considered ‘polynomial’ (where $\ell(\cdot) := |\varphi|$ for operator $\varphi$) — namely second-order polynomial [6], [1], [2], that is, any term composed from $n$, $\ell(\cdot)$, $+$, $\times$, and positive integer constants.

II. REPRESENTING POWER SERIES

As already mentioned [8] a power series needs, in addition to the coefficient sequence $(a_j)_j$, further information to admit computable evaluation on a given argument $x$. [10, Thm. 4.3.11] Thus, one possible representation of $f$ (analytic on $\mathbb{B}(0,1)$) by encoding its power series around $z_0 = 0$

Definition 5. Let $\alpha \subseteq \{0,1\}^\omega \rightarrow C^\omega(\mathbb{B}(0,1))$ denote a fiber product representation according to [10, Def. 3.3.3+3.3.7], informally described as follows:

An $\alpha$-name of $f : \mathbb{B}(0,1) \ni z \mapsto \sum a_j z^j$ of a $(p_{\alpha,j})^\omega$-name of the sequence $(a_j)_j \subseteq \mathbb{C}$, an $\nu$-name of some integer upper bound $K$ on $1/\log_2 R$ and a $\nu_\alpha$-name of some integer $A$ satisfying $|a_j| \leq A/r^j$ for all $j$ with $r : = \sqrt[4]{2} < R$.

Here $R > 1$ denotes the radius of convergence of $f$ around 0.

Theorem 6. a) Evaluation $C^\omega(\mathbb{B}(0,1)) \times \mathbb{B}(0,1) \ni (f, z) \mapsto f(z) \in \mathbb{C}$ is $(\alpha, \rho_{\alpha}, p_{\rho_{\alpha}})$-computable within time polynomial in $K + \log A + n$.

b) Addition and multiplication $C^\omega(\mathbb{B}(0,1))^2 \ni (f, g) \mapsto f + g, f \cdot g \in C^\omega(\mathbb{B}(0,1))$ is $(\alpha, \alpha, \alpha)$-computable within time poly. in $K + \log A + n$.

c) Differentiation and anti-differentiation:

$C^\omega(\mathbb{B}(0,1)) \ni (f, z) \mapsto f', \int f \in C^\omega(\mathbb{B}(0,1))$: (\alpha, \alpha)$-computable within time poly. in $K + \log B + n$.

d) Maximization on $[-1,1]$, that is $C^\omega(\mathbb{B}(0,1)) \ni f \mapsto \max\{|f(x)| : -1 \leq x \leq 1\} \in \mathbb{R}$, is $(\alpha, \rho_{\alpha})$-computable within time polynomial in $K + \log A + n$.

e) Given $f|_{[-1,1]}$ as an oracle (Sec. 1-B) and $K \geq 1/\log_2 R$ as well as an integer upper bound $B$ on $\max\{|f(z)| : |z| \leq \sqrt[4]{2}\}$, an $\alpha$-name of $f$ is computable within time polynomial in $K + \log B + n$.

We emphasize that the algorithms implicitly described within the respective proofs of a) to e) are actually practical and ready to implement in, say, $\text{IRRAM}$ [9].

III. REPRESENTING REAL ANALYTIC FUNCTIONS

We focus on real functions $f : \{0,1\} \rightarrow \mathbb{R}$ analytic on some complex neighbourhood of $[0,1]$. Being members of $C\{0,1\}$, these are however already equipped with the above second-order representation $[p_0 \rightarrow p_0]$. In order to compare both we enrich it with some additional information and turn the above $\alpha$ into a second-order representation:

Definition 7. a) Let $C^\omega([0,1])$ denote the space of functions analytic on some complex neighbourhood of $[0,1]$.

b) Let a $\tilde{\beta}$-name of such $f$ consist of

• a $p_0 \rightarrow p_0$-name of $f|_{[0,1]}$, together with

• an integer $L$ in unary such that $f$ is complex analytic even on (an open neighbourhood of) the closed rectangle $R_L := \{x + iy : \frac{-1}{2} \leq y \leq \frac{1}{2}, \frac{-1}{2} \leq x \leq \frac{L+1}{2}\}$

• and a binary integer upper bound $B$ on $|f|$ on said $R_L$.

c) Let $M, A_m, L_m \in \mathbb{N}$, $x_m \in [0,1]$ and $a_{m,j} \in \mathbb{R}$ for $1 \leq m \leq M$. We say that $(M, (x_m), (a_{m,j}), (L_m), (A_m))$ represents $f \in C^\omega([0,1])$ if it holds

\[
\{0,1\} \subseteq \bigcup_{n=1}^{L} [x_m - \frac{4}{2^{n-1}2^{m}}, x_m + \frac{4}{2^{n-1}2^{m}}],
\]

f(0)(x_m) = a_{m,1} \cdot j, \text{ and } \max\{|a_{m,j}| \leq A_m \cdot L_m\}.

An $\tilde{\alpha}$-name of $f$ in $C^\omega([0,1])$ encodes, according to [10, Def. 3.3.7], the following: a $p_{\tilde{\alpha},d}$-name of $x_m$, a $p_{\tilde{\alpha},u}$-name of $\{a_{m,j}\}$, and a $\nu_{\tilde{\alpha}}$-name of $A_m$ (i.e. in binary) as well $L_m$ in unary.

Using $\tilde{\alpha}$ and $\tilde{\beta}$, the (in Rem. 3) advertised dependence of the complexity on additional (encoded) parameters follows:

Theorem 8. a) On $C^\omega([0,1])$, $\tilde{\alpha}$ and $\tilde{\beta}$ constitute second-order polytime equivalent representations.

b) Evaluation, addition, multiplication, anti-differentiation, and maximization (cf. Thm. 6) on $C^\omega([0,1])$ are uniformly second-order polytime computable.

IV. CONCLUSION AND PERSPECTIVES

The literature on TTE provides some categorical constructions of natural representations for certain spaces — and can prove them optimal. Strengthening from computability to complexity, Theorem 8a) shows two 2nd-order representations of the space $C^\omega([0,1])$ of real analytic functions on $[0,1]$ to be 2nd-order polytime equivalent and render the basic primitives second-order polytime computable. This subsumes several known nonuniform results.

REFERENCES


A Metric Analogue of Stone Duality for Markov Processes

Kim Larsen  
Department of Computer Science  
University of Aalborg  
Aalborg, Denmark

Radu Mardare  
Department of Computer Science  
University of Aalborg  
Aalborg, Denmark

Prakash Panangaden  
School of Computer Science  
McGill University  
Montreal, Canada

I. INTRODUCTION

The Stone representation [7] theorem is one of the recognized landmarks of mathematics. The Stone representation theorem [7] states that every (abstract) boolean algebra is isomorphic to a boolean algebra of sets; in modern terminology one has an equivalence of categories between the category of boolean algebras and the (opposite of) the category of compact Hausdorff zero-dimensional spaces, or Stone spaces.

In this paper we develop exactly such a duality for continuous-time continuous-space transitions systems where transitions are governed by an exponentially-distributed waiting time, essentially a continuous-time Markov chain (CTMC) with a continuous space. The logical characterization of bisimulation for such systems was proved a few years ago [3] using much the same techniques as were used for labelled Markov processes [5]. Recent work by the first two authors and Cardelli [1], [2] have established completeness theorems and finite model theorems for similar logics. Thus it seemed ripe to capture these logics algebraically and explore duality theory.

One of the critiques of logics and equivalences being used for the treatment of probabilistic systems is that boolean logic is not robust with respect to small perturbations of the real-valued system parameters. Accordingly, a theory of metrics [4] was developed and metric reasoning principles were advocated. In conjunction with our exploration of duality theory therefore we investigated the role of metrics and discovered a striking metric analogue of the duality theory. This paper describes both these theories. One can view the latter as the analogue of a completeness theorem for metric reasoning principles.

One of the points of departure of the present work from earlier work is the use of hemimetrics: analogues of pseudometrics that are not symmetric. This fits in well with the order structure of the boolean algebra. Nearly 25 years ago, Mike Smyth [6] advocated the use of such structure to combine metric and domain theory ideas. The interplay between the hemimetric and the boolean algebra is somewhat delicate and had to be carefully examined for the duality to emerge. It is a pleasant feature that exactly these axioms relating the hemimetric and the boolean algebra are satisfied in our examples without any artificial fiddling.

We summarize the key results of the present work:

- a description of a new class of algebras that captures, in algebraic form, the probabilistic modal logics used for continuous Markov processes,
- a duality between these algebras and continuous Markov processes
- a (hemi)metrized version of the algebras and of the Markov processes and
- a metric analogue of the duality.

II. DEFINITIONS

Let \( M \) be a set and \( d : M \times M \rightarrow \mathbb{R} \).

**Definition 1.** We say that \( d \) is a hemimetric on \( M \) if for arbitrary \( x, y \in M \),

\[
(1): \quad d(x, x) = 0 \\
(2): \quad d(x, y) \leq d(x, z) + d(z, y)
\]

We say that \((M, d)\) is a hemimetric space.

Note that a hemimetric is not necessarily symmetric nor does \( d(x, y) = 0 \) imply that \( x = y \). A symmetric hemimetric is called a pseudometric.

**Definition 2.** For a hemimetric \( d \) on \( M \) we define the Hausdorff hemimetric \( d^H \) on the class of subsets of \( X \) by

\[
d^H(X, Y) = \sup_{x \in X} \inf_{y \in Y} d(x, y).
\]

We also define the dual of the Hausdorff hemimetric \( d^H \) on the class of subsets of \( X \) by

\[
d^H(X, Y) = \sup_{y \in Y} \inf_{x \in X} d(x, y).
\]

**Definition 3** (Continuous Markov processes). Given a measurable space \((M, \Sigma)\), a continuous Markov process (CMP) is a tuple \( M = (M, \Sigma, \theta) \), where \( \theta \in \mathbb{M} (M \mapsto \Delta (M, \Sigma)) \). \( M \) is the support set of \( M \) denoted by \( \text{supp}(M) \). If \( m \in M \), \((M, m)\) is a continuous Markov process (CMP).

**Definition 4** (Aumann algebra). An Aumann algebra \((AA)\) over the set \( B \neq \emptyset \) is a structure \( A = (B, \tau, \bot, \cup, \cap, \{F \mapsto G \}_{r \in \mathbb{Q}^+}, \mathbb{E}) \) where \( B = (B, \tau, \bot, \cup, \cap, \mathbb{E}) \) is a meet-continuous boolean Algebra, for each \( r \in \mathbb{Q}^+ \), \( F_r, G_r : B \rightarrow B \) are monadic operations and the sets of \( B \) satisfy the axioms in Table 1, for arbitrary \( a, b \in B \) and \( r, s \in \mathbb{Q}^+ \).
Definition 5 (Metrized Aumann algebra). A metrized Aumann algebra is a tuple \((\mathcal{A}, \delta)\), where \(\mathcal{A} = (B, T, \bot, \sim, \sqcup, \sqcap, \{F_r, G_r\}_{r \in \mathbb{Q}_+}, \sqsubseteq)\) is an Aumann algebra and \(\delta : B \times B \rightarrow [0, 1]\) is a hemimetric on \(B\) satisfying, for arbitrary \(a, b \in B\), and arbitrary filtered set \(A \subseteq B\) for which there exists \(\bigwedge A' \in B\), the axioms in Table II.

\[
\begin{align*}
(H0): & \quad \delta(a, b) = 0, \text{ then } a \sqsubseteq b \\
(H1): & \quad \delta(a, b) = \delta(a \cap(-b), b) \\
(H2): & \quad \delta(b, a \cap A) = \inf_{a \in A} \delta(b, a) \\
(H3): & \quad \delta(\bigwedge A, b) = \sup_{a \in A} \delta(a, b)
\end{align*}
\]

**Table II**

Hemimetric axioms for metrized AA

We have a duality theorem between CMPs and Aumann Algebras.

**Theorem 6** (Representation Theorem). (i) Any CMP \(M = (\mathcal{M}, \Sigma, \delta)\) is bisimilar to \(\mathcal{M}(\mathcal{L}(\mathcal{M}))\) and the bisimulation relation is given by the mapping \(\alpha\) defined, for arbitrary \(m \in M\), by

\[
m \mapsto \alpha(m) = \{\phi \in \mathcal{L}(\mathcal{M}) \mid M, m \models \phi\}.
\]

(ii) Any Aumann algebra \(\mathcal{A} = (B, T, \bot, \sim, \sqcup, \sqcap, \{F_r, G_r\}_{r \in \mathbb{Q}_+}, \sqsubseteq)\) is isomorphic to \(\mathcal{L}(\mathcal{M}(\mathcal{A}))\) and the isomorphism is given by the mapping \(\beta\) defined, for arbitrary \(a \in B\), by

\[
a \mapsto \beta(a) = \bigwedge \{\phi \in \mathcal{L}(\mathcal{M}(\mathcal{A})) \mid \forall u \in \mathcal{U}(B) \text{ s. t. } (a) \sqsubseteq u, \mathcal{M}(\mathcal{A}), u \models \phi\}.
\]

This extends to a duality between the hemi-metric spaces in the following sense.

**Theorem 7** (The metric duality theorem). (i) Given a metrized CMP \((M, d)\) with \(M = (\mathcal{M}, \Sigma, \delta)\), \(M\) is bisimilar to \(\mathcal{M}(\mathcal{L}(\mathcal{M}))\) by the map \(\alpha\) defined in the Representation Theorem and, in addition, for arbitrary \(m, n \in M\),

\[
d(m, n) = (d^M)_H(\alpha(m), \alpha(n)).
\]

(ii) Given a metrized AA \((\mathcal{A}, \delta)\) with \(\mathcal{A} = (B, T, \bot, \sim, \sqcup, \sqcap, \{F_r, G_r\}_{r \in \mathbb{Q}_+}, \sqsubseteq)\), \(\mathcal{A}\) is isomorphic to \(\mathcal{A}(\mathcal{L}(\mathcal{M}(\mathcal{A}))\) by the map \(\beta\) defined in the Representation Theorem and, in addition, for arbitrary \(a, b \in B\)

\[
\delta(a, b) = (d^A)_H(\beta(a), \beta(b)).
\]
The Comprehension Cube
and its Primitive-Recursive Corner

Daniel Leivant, Indiana University and PPS Université Paris-Diderot

Abstract—The set-existence (comprehension) principle of second-order logic admits all formulas as set definitions, notwithstanding their potential implicit circularity. Three well-studied computationally-related approaches to restrict circularity are the restriction of set-existence to computational (i.e. strict-$\Pi^1_1$) formulas, scope-separation of quantifiers, and ramification. These are independent, giving rise to a cube of eight combinations. Each deductive formalism of (pure) second-order logic naturally spawns its provably-computable functions. While the functions corresponding to computational comprehension are the provable recursive functions of arithmetic, we show that a further restriction to scope-separated formulas yields the primitive-recursive functions.

I. FUNCTION PROVABILITY IN SECOND-ORDER LOGIC

The classification outlined in the Abstract is made possible in the setting of [8] for studying computational complexity within pure second-order logic. It is based on two century-old observations:

1) Each inductive type, such as words and natural numbers, can be construed as a free algebra $\mathcal{A}(C)$ over some finite set $C$ of constructors.

2) Each such type can be delineated by a formula using a quantifier over sets:

$$D_C[x] \equiv \forall S \text{Cl}_C[S] \rightarrow S(x)$$

where $\text{Cl}_C[S]$ states that the set-variable $S$ is closed under the constructors.

Our programming paradigm of choice is equational programs, since it meshes well with the syntax, semantic, and deductive aspects of pure logic. Beyond constructors one uses variables as well as program-function identifiers. A program-equation is a phrase of the form $g(t) = e$, where $g$ is a program-function of some arity $r$, $t$ are $r$ base-terms, and $e$ is a program-term. An equational-program $(P, f)$ (or arity $r$) consists of a finite set $P$ of pairwise non-unifiable program-equations, and a program-function $f$ (or arity $r$) singled out as principal identifier. A program $(P, f)$ computes a partial-function $g$ over the set $D_C$ of data-terms when $g(t) = q$ just in case the equation $f(t) = q$ is derivable in equational logic from the equations of $P$.

We say that a program $(P, f)$ of arity $r$, over $C$, is provable in a second-order logic $L$ if

$$\forall \exists g \in D_C(z) \rightarrow D_C(f(z))$$

is derivable in $L$ from the universal closure of $P$. A function $g$ over $\mathcal{A}(C)$ is provable in $L$ if it is computed by a program provable in $L$. (We focus on second-order logic with quantification over relations, not functions.) The following is an easy corollary of Prawitz’s interpretation of second-order arithmetic in second-order logic [10]:

THEOREM 1: A function is provable in full second-order logic iff it is a provably-recursive function of second-order arithmetic.

II. COMPUTATIONAL SET-EXISTENCE

We write $L_2$ for full second-order logic (based on relational quantifiers), and if $C$ is a class of formulas, then $L_2(C)$ is $L_2$ with comprehension restricted to formulas in $C$.

A second-order formula $\varphi$ is computational if all relational-quantifiers therein are positively-occurring $\forall$ and all object-quantifiers are positively occurring $\exists$. Write Comp for the set of computational formulas.

Computational formulas are equivalent (modulo a weak choice principle) to Kreisel’s strict-$\Pi^1_1$ formulas [6]. They define over $\mathbb{N}$ no more than the RE relations, and arguably capture the notion of computability over arbitrary structures (see e.g. [5], [7]). Moreover the latter fact can be proved in a weak variant of second order arithmetic, with only recursive comprehension, $\Sigma^0_1$-induction, and Weak König’s Lemma, a variant whose provably recursive functions are all primitive recursive (see e.g. [4] or [11]).

III. VARIABLE-SEPARATED COMPREHENSION

A formula $\varphi$ is separated if every subformula of the form $\forall R \psi$ or $\exists R \psi$ has $R$ as the only free relational variable in $\psi$. Thus every two relational quantifiers in $\varphi$ are “separated” in that one does not reach into the scope of the other. Separated formulas, also dubbed “non-interleaving,” were studied, in a type-theoretic guise, by Aehlig [1], [2], [3], who identified the complexity of separated comprehension (in a type-theoretic
guise) to be that of the theory $\text{ID}_\omega$ if finitely-iterated inductive definitions [9].

This implies quite easily

**THEOREM 2:** (Essentially Aehlig’s) The functions provable in second-order logic with comprehension restricted to separated formulas are precisely the provably-recursive functions of $\text{ID}_\omega$.

Let $\text{SepComp}$ be the set of scope-separated computational formulas.

**PROPOSITION 3:** Every primitive-recursive function is provable in $L_2(\text{SepComp})$.

**Proof Outline.** It is an easy exercise to show that if $g_0$ and $g_s$ are provable in $L_2(\text{SepComp})$, then so is the function $f$ defined from them by recurrence:

\[ f(0, \overline{y}) = g_0(\overline{y}) \quad f(x + 1, \overline{y}) = g_s(x, \overline{y}, f(x, \overline{y})) \]

The Proposition falls out then by a discourse-level induction on the generative definition of primitive-recursive functions.

**IV. PROVABLE FUNCTIONS ARE PRIMITIVE RECURSIVE**

We prove next the converse of Proposition 3, thereby establishing our main result:

**THEOREM 4:** A function is provable in $L_2(\text{SepComp})$ iff it is primitive-recursive.

We demonstrate Theorem 4 using standard proof theoretic tools, which we assume to be familiar.

**LEMMA 5:** Fix $k \geq 0$. The Normalization Theorem for natural deductions of $L_2(\text{SepComp})$ with $\leq k$ nested comprehensions is provable in Primitive Recursive Arithmetic.

**Proof Outline.** We consider Girard’s proof for the natural-deduction calculus in second-order logic. When deductions use only $\text{SepComp}$ comprehension, the normalization proof can itself be formalized in second-order arithmetic with comprehension restricted to separated computational formulas of second-order arithmetic (where we assume function-identifiers for all primitive-recursive functions). More precisely, the Normalization Theorem for the collection of derivations with $\leq k$ nested instances of $\text{SepComp}$-comprehension can be proved in second-order arithmetic with $\leq k$ instances of $\text{SepComp}$-comprehension.

Consider the theory $\text{WKL}_0$ of second-order arithmetic with Comprehension for recursive formulas, Induction for $\Sigma^0_1$ formulas, and with Weak König’s Lemma (WKL) as an axiom (stating that every infinite binary tree has an infinite branch); see e.g. [11], [4]. We can prove within $\text{WKL}_0$ that every separated computational formula in the language of arithmetic is equivalent to a $\Sigma^0_1$ formula. Thus the Normalization Theorem for the collection of derivations with $\leq k$ nested instances of $\text{SepComp}$-comprehension is provable in $\text{WKL}_0$. But the Normalization Theorem is a $\Pi^0_1$ formula (“every eduction starts some reduction sequence leading to a normal derivation”), and since $\text{WKL}_0$ is conservative over Primitive Recursive Arithmetic with respect to $\Pi^0_1$ [11], [4], the Lemma follows.

**Proof Outline of the backward implication of Theorem 4.** Let $(\mathcal{P}, \Gamma)$ be a program computing a function $g : \mathbb{N} \rightarrow \mathbb{N}$, and let $\Delta$ be a derivation of $\forall x \exists y f(x, y)$ from $\mathcal{P}$ in $L_2(\text{SepComp})$, with $\leq k$ nested instances of $\text{SepComp}$-comprehension. By Lemma 5 it follows that Primitive Recursive Arithmetic proves that for each $n \geq 0$ there is a normal derivation $\Pi_n$ of $\forall \overline{n} \exists \overline{y} f(\overline{n}, \overline{y})$ from $\mathcal{P}$. So there is a primitive recursive function that yields for input $n$ the derivation $\Pi_n$ (suitably coded). Since from $\Pi_n$ we can easily extract the value of $g(n)$, it follows that $g$ is primitive-recursive.

**V. COMPUTATIONAL COMPREHENSION, UNSEPARATED**

When comprehension formulas are computational, but not necessarily variable-separated, we obtain a far larger class of functions:

**THEOREM 6:** A function $f$ is provably recursive in first-order (Peano) Arithmetic, that is, $f$ is definable by recursion over some ordinal $\alpha < \varepsilon_0$.

The forward direction of Theorem 6 is analogous to that of Lemma 5, except that the nesting-depth of comprehension corresponds to levels in the Arithmetical Hierarchy of the induction formulas.

**Acknowledgment.** This work was also supported by LORIA Nancy.

**REFERENCES**


On the Expressivity of Linear Logic Subsystems
Characterizing Polynomial Time

Matthieu Perrinel
Laboratoire d’Informatique du Parallélisme
ENS Lyon – Université de Lyon UMR 5668 CNRS ENS-Lyon UCBL INRIA
46, allée d’Italie 69364 Lyon cedex 07 – FRANCE

Abstract—Implicit computational complexity characterizes complexity classes by restricting models of computation (lambda calculus, recursion theory, linear logic,...). All previously studied subsystems of linear logic characterizing Ptime lack expressive power. Although every Ptime function can be computed in those systems, they do not always accept the most usual algorithms. In this work we will approach the “expressivity frontier”: the sets of programs that could be accepted by Ptime sound linear logic subsystem.

I. INTRODUCTION

a) Implicit computational complexity: For decades, computer scientists have tried to characterize classes of complexity by restricting syntactically models of computation. This effort is known as implicit computational complexity. The main goal is to achieve automated certification of the complexity of a program. Some consider also other goals for implicit computational complexity: to understand the roots of complexity, as suggested by Girard [3], or to create polytime mathematics, as suggested by Girard [3].

Many characterizations of polynomial time have been found for different models of computation: restriction of recursion [2], non-size-increasing computation [5], quasi-interpretations on a functional language and restriction of linear logic [6] [3] [1]. Each approach has its own assets and drawbacks with respect to our goal of statically inferring bounds on real-life programs.

b) Linear logic and expressivity: One of the interests of the linear logic approach is the possibility to quantify over types (second order types). This allows us, for example, to write a sorting algorithm which can be applied to any type of data, as soon as a comparison function is given. The first really implicit characterization of polynomial time by a linear logic subsystem was Light Linear Logic (LLL) by Girard [3].

The main drawback of LLL is the lack of expressivity. A completeness result shows that any Turing machine terminating in polytime can be simulated by a LLL proof-net (proof-nets are the programs of linear logic). However we cannot expect programmers to stick to the simulation of Turing machines. Therefore, we are more interested in the programming of high-level algorithms, but many of those are not typable in LLL. And many of those algorithms are not typable in LLL (for example the usual multiplication on Church integers and quick-sort). Among the other linear logic characterizations of Ptime, SLL [6] is even less expressive than LLL. L^4 and L^4_0 [1] generalize LLL, but the increase in expressivity is very limited.

Therefore, we can wonder if the lack of expressivity is a drawback of the current subsystems (SLL, BLL, LLL, L^4, L^4_0) or a limitation inherent to the linear logic approach? If there are usual algorithms which are not accepted by any Ptime linear logic subsystem, then it would raise doubts on the usability of the linear logic approach for real life applications. In this work, we give a sufficient criterium for Ptime which we think is broader than previous criteria. We also give a sufficient criterium for elementary time (towers of exponentials of fixed height), and a necessary criterium for elementary time, related to the sufficient criterium. Those criteria are based on Dal Lago’s context semantics [7].

c) Context semantics: The idea of geometry of interaction [4] and context semantics is to study the reduction of proof-nets (or λ-terms) by leaving the proof-net unchanged and analysing instead some paths in it. Context semantics has first been used to study qualitative properties. In [7], Ugo Dal Lago adapted context semantics to study quantitative properties of cut-reduction. From this point of view an advantage of context semantics compared to the syntactic study of reduction is its genericity: some common results can be proven for different variants of linear logic, which allows to factor out proofs of complexity results for these various systems. Indeed strong bounds were proven in context semantics for SLL, ELL, LLL, and L^4.

A side effect of this proof factorization is the emergence of a common structure in those proofs, and gives a new point of view on them. Context semantics gives semantics characterization of the “stratification” condition and the “dependence control” condition pointed out by Baillot and Mazza [1].

II. CHARACTERIZATION OF STRATIFICATION AND DEPENDENCE CONTROL

d) Context semantics characterization of stratification: The idea of stratification is to forbid self-duplication. We define a duplication relation on boxes and we define stratified proof nets as the proof nets whose duplication relation is acyclic.

In the following definition, σ(B) refers to the edge leaving the principal door of box B. σ_i(B) refers to the edge leaving the i-th auxiliary door of box B.
a \downarrow A \xrightarrow{b} \frac{c}{B} \\
\vphantom{A^a} A \otimes B

\begin{array}{l}
(a, U, V, +) \Rightarrow (e, U, U !_1 :: V, +) \\
(b, U, V, +) \Rightarrow (e, U, U !_2 :: V, +)
\end{array}

\begin{array}{l}
\epsilon_1 \downarrow \frac{c_2}{C} \\
\vphantom{A^a} ? A \ \vphantom{C} ? A
\end{array}

\begin{array}{l}
(e_i, U, ? :: V, +) \Rightarrow (g, U, ? :: V, +)
\end{array}

\begin{array}{l}
\epsilon \downarrow A \\
\vphantom{A^a} ? P \ \vphantom{A} \ \vphantom{A} \vphantom{A} g \ A \ \vphantom{A} \vphantom{A}
\end{array}

\begin{array}{l}
\epsilon_i \downarrow \frac{g_i}{C} \\
\vphantom{A^a} ? A \ \vphantom{C} ? A
\end{array}

\begin{array}{l}
(e, t \circ U, ? :: V, +) \Rightarrow (f, U, ? :: V, +) \\
(g, t \circ U, ? :: V, +) \Rightarrow (h, U, ? :: V, +) \\
(f, U, ? :: V, +) \Rightarrow (h, U, ? :: V, +)
\end{array}

---

**Definition 1** \( (\Rightarrow) \). "B is duplicated directly by \( B' \) " is defined by:

\[ B \Rightarrow B' \Leftrightarrow \exists U, V, \exists t, (\sigma(B), U, !_1, +) \Rightarrow_G (\sigma(B'), V, +) \]

Some rules defining \( \Rightarrow_G \) are given on Fig. 1.

**Definition 2** (Stratification). A proof net is stratified if \( \Rightarrow \) is acyclic.

**Theorem 1.** A stratified proof net reduces in a number of steps bounded by an elementary function of its size. The elementary function only depends on the depth of \( \Rightarrow \) and the depth of the boxes.

ELL and L³ proof nets are stratified. So this theorem directly gives us a strong elementary bound for those systems.

e) Context semantics characterization of dependence control:

**Definition 3.** "B depends directly on \( B' \) " is defined by:

\[ B \Rightarrow B' \Leftrightarrow \exists U, V, \exists t, (\sigma(B), U, !_1, +) \Rightarrow_G (\sigma(B'), V, +) \]

**Definition 4.** A proof net controls dependence if each box belongs to at most one \( \Rightarrow \) cycle.

**Theorem 2.** The maximal reduction length of a stratified proof net which controls dependence is a polynomial of its size. The degree of the polynomial only depends on the depth of \( \Rightarrow \) and \( \Rightarrow \) and the depth of the proof-net.

---

**III. TOWARDS A NECESSARY CONDITION**

Even if the criteria we found seem very broad, we do not know if we could have even broader conditions or not. In order to see if there is still much room for improvement, we proved a necessary criterion for proof-nets to be elementary time.

**Theorem 3.** Let \( \pi_0 \) be a cut-free proof-net and \( n \) be a cut-free proof-net representing a natural number. And let \( \pi \) be the proof-net representing the application of \( \pi_0 \) to \( n \). Suppose that:

1) There exists a box \( B \) such that \( B \Rightarrow B \) (\( \pi \) is not stratified).
2) The path involved in condition 1 goes through a box \( B' \) \( \Rightarrow B' \) (\( \pi \) does not satisfy dependence control).
3) Let \( U \) and \( V \) be the potentials such that \( (B, U) \Rightarrow (B, V) \). We suppose that the transformation from \( U \) to \( V \) is only done in \( n \).

Then the execution time of \( \pi_0 \) applied to another integer \( k \) can not be bounded by an elementary function on \( k \).

Ideally, we would like a theorem saying that if a “reasonable” system accepts a non-stratified proof-net, then it also accepts a non-elementary proof-net. Here, we need some strong side conditions. So, the result is still incomplete. However, it is the first time to our knowledge, that a sufficient condition and a necessary condition have been linked for a complexity class, in a linear logic setting. This result can be thought of as a guide for finding more expressive systems. If we want a system which is elementary sound and which does not always satisfy stratification, then the system must break one of the side conditions when it breaks stratification.

---

**IV. CONCLUSION**

Our sufficient conditions allow further refactoring of the proofs of strong bounds for light linear logic subsystems. For the systems whose strong bound has already been proved, we gain no direct knowledge. However, we think that it can make proofs easier to understand. For systems whose strong bound had not been proved yet (L³ and L³⁺ for example), it eased significantly the search for a proof.

The necessary condition gives us a better understanding on how expressive a system could be. However, there is still a big gap between the sufficient and the necessary conditions for elementary time. Bridging this gap as much as possible is our current priority.

The new insight given by the conditions gave ideas for more expressive systems, where the levels would be on the exponential connectives of formulas (and not on edges).

**REFERENCES**
