Invited talks

An infinitary treatment of temporal logic

Bahareh Afshari

1st February 2019

Abstract

We explore the proof theory of fixed point modal logic with converse modalities, commonly known as 'full mu-calculus'. Building on nested sequent calculi for tense logics [2] and infinitary proof theory of fixed point logics [1], a cut-free sound and complete proof system for full mu-calculus is proposed. As a corollary of our framework, we also obtain a direct proof of the regular model property for the logic [4]: every satisfiable formula has a tree model with finitely many distinct subtrees. To obtain this result we appeal to the basic theory of well-quasi-orderings in the spirit of Kozen's proof of the finite model property for μ-calculus [3].

References

[1] G. Jäger, M. Kretz and T. Studer. "Canonical completeness of infinitary mu". In: *J. Log. Algebr. Program.* 76.2 (2008), pp. 270–292.

[2] R. Kashima. "Cut-free sequent calculi for some tense logics". In: *Studia Logica: An International Journal for Symbolic Logic* 53.1 (1994), pp. 119–135.

[3] D. Kozen. "Results on the propositional mu-calculus". In: *Theor. Comput. Sci.* 27 (1983), pp. 333–354.

[4] M. Y. Vardi. "Reasoning about the past with two-way automata". In: *Automata, Languages and Programming*. Ed. by K. G. Larsen, S. Skyum and G. Winskel. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 628–641.

Circular (Yet Sound) Proofs

Albert Atserias
Universitat Politècnica de Catalunya
atserias@lsi.upc.edu

We introduce a new way of composing proofs in rule-based proof systems that generalizes tree-like and dag-like proofs. In the new definition, proofs are directed graphs of derived formulas, in which cycles are allowed as long as every formula is derived at least as many times as it is required as a premise. We call such proofs circular and we show that, for all sets of standard inference rules, circular proofs are sound. We first focus on the circular version of Resolution, and we immediately see that it is stronger than Resolution since, as we show, the pigeonhole principle has circular Resolution proofs of polynomial size. Surprisingly, as proof systems for deriving clauses from clauses, Circular Resolution turns out to be equivalent to Sherali-Adams, a proof system for reasoning through polynomial inequalities that has linear programming at its base. As corollaries we get: 1) polynomial-time (LP-based) algorithms that find circular Resolution proofs of constant width, 2) examples that separate circular from dag-like Resolution, such as the pigeonhole principle and its variants, and 3) exponentially hard cases for circular Resolution. Contrary to the case of circular resolution, for Frege we show that circular proofs can be converted into tree-like ones with at most polynomial overhead.

Finiteness properties of simple types through a coinductive lambda-calculus

Ralph Matthes
CNRS, Institut de Recherche en Informatique de Toulouse
ralph.matthes@irit.fr

Proofs in propositional logic correspond to lambda-terms with simple types. The search for proofs of a given formula/type corresponds to the search for lambda-terms of that type. Finite successful runs of the search correspond to the construction of inhabitants of that type, but proof search is more than the set of its successful outcomes.

In joint work with José Espírito Santo and Luís Pinto – the whole talk is about joint work with them – we developed a representation of the search space for locally correct applications of the proof rules, not limiting the representation to the construction of (finite) inhabitants. The data structure we propose is an extension of lambda-calculus (restricted to long normal forms) to a coinductive structure that also allows to express choice points in the search process. The elements of that structure are called forests, and individual inductive or even coinductive lambda-terms can be seen as members of such a forest.

The forests form a coinductive datatype and are therefore not necessarily finitely described objects. The finitary counterpart to forests is a lambda-calculus with inductively defined terms (called finitary forests) that also has the means of expressing choice points and that comes with a formal fixed-point operator, based on fixed-point variables that are typed with sequents.

The methodology suggested by these structures is to specify proof search problems through forests and by solving them effectively through finitary forests.

In particular, we used this to reprove decidability of inhabitation and type finiteness (i. e., the property of having only finitely many inhabitants), but also the prediction of at most one inhabitant or type finiteness based on the absence of two atom occurrences at negative resp. positive position in the formula.

We now come to as of now unreleased material. Type finiteness "naturally" means having only finitely many inhabitants. But, as soon as one grants the status of "member" of a formula A to any member of the forest canonically generated from search for proofs of A, other natural concepts of finiteness are possible. One is the finiteness of any member of A. This concept may be related to the finiteness of the whole search space, hence a third concept of finiteness, in the spirit of König's lemma. Our main new result is that all these concepts of finiteness are decidable, and so is the property of absence of members (finite

or otherwise), which is an extreme form of unprovability.

The interesting technical observation here is that all three finiteness concepts are instances of a generic finiteness predicate with a set parameter, and their decidability is seen by a single proof for that predicate.

Time permitting, we will also discuss "pruning" of the search space generated from a formula, where all failed runs are chopped off. Our final result, which we dub König's lemma for simple types, says that finiteness of all members of a formula is equivalent to finiteness of the pruned search space generated from the formula.

Provability and consistency of circuit lower bounds

Moritz Mueller
Universitat Politècnica de Catalunya
moritz@cs.upc.edu

In 1995 Razborov asked for the right fragment of bounded arithmetic capturing existing techniques to prove circuit lower bounds for explicit Boolean functions. The talk reports some new developments.

Systems for constructive reverse mathematics

Takako Nemoto

In the reverse mathematics over classical logic, most of the results are given on subsystems of Z_2, whose language is for natural numbers and sets of them, besides the works on higher order reverse mathematics such as [8] and reverse recursion theory such as [4].

In the reverse mathematics over intuitionistic logic has not yet had fixed language and systems. Here are some examples:

- **Unformalized reverse mathematics**: They do not use any formal systems and aiming to classify mathematical theorems by the equivalence over Bishop style constructive mathematics [3]. ([2], [5], etc..)
- **Reverse mathematics with function based language**: They use the language for natural numbers and functions over them. Depending on the treatment of function symbols, there are many variants. ([1], [6], [9], [11], etc..)
- **Reverse mathematics with higher order arithmetic**: They use the language for higher order arithmetic. ([7], etc..)
- **Reverse mathematics with first order arithmetic**: They use the language for first order arithmetic. ([10], etc..)

In this talk, we consider the relationship among many systems for constructive reverse mathematics from interpretability and conservativity. We also consider the relationship with systems over classical logic.

References

[1] J. Berger, H. Ishihara, T. Kihara and T. Nemoto, *The binary expansion and the intermediate value theorem in constructive reverse mathematics*, Arch. Math. Logic 58 (2018), pp. 203217. https://doi.org/10.1007/s00153-018-0627-2

[2] J. Berger and G. Svindland, *A separating hyperplane theorem, the fundamental theorem of asset pricing, and Markov' s principle*, Ann. Pure Appl. Logic, 167 (2016), pp. 1161-1170.

[3] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, New York (1967).

[4] C. T. Chong, Wei Li and Yue Yang, *Nonstandard models in recursion theory and reverse mathematics*, Bulletin of Symbolic Logic 20 (2014), pp. 170-200.

[5] H. Diener, *Constructive reverse mathematics*, Habilitation Schrift, Universtät Siegen (2018).

[6] H. Ishihara, *Constructive Reverse Mathematics: Compactness Properties*, In: P. Schuster and L. Crosilla eds., From Sets and Types to Topology and Analysis: Towards practicable foundations for constructive mathematics, Oxford University Press (2005), pp. 245-267.

[7] H. Ishihara, *On Brouwer's continuity principle*, Indagationes Mathematicae 29 (2018), pp. 1511-1524.

[8] U. Kohlenbach, *Higher order reverse mathematics*, Reverse mathematics 2001, Lect. Notes Log., 21, Assoc. Symbol. Logic, La Jolla, CA (2005), pp. 281295.

[9] I. Loeb, *Equivalents of the (Weak) Fan Theorem*, Annals of Pure and Applied Logic, Volume 132, Issue 1, 2005, pp. 51-66.

[10] T. Nemoto, *Finite sets and infinite sets in weak intuitionistic arithmetic*, submitted for publication.

[11] W. Veldman, *Brouwer's fan theorem as an axiom and as a contrast to Kleene's alternative*, Arch. Math. Logic 53 (2014), pp. 621-693.

Ramsey-like theorems and moduli of computation

Ludovic Patey
Institut Camille Jordan
ludovic.patey@computability.fr

Ramsey's theorem asserts that every k-coloring of $[\omega]^n$ admits an infinite monochromatic set. Whenever $n \geq 3$, there exists a computable k-coloring of $[\omega]^n$ whose solutions compute the halting set. On the other hand, for every computable k-coloring of $[\omega]^2$ and every non-computable set C, there is an infinite monochromatic set H such that $C \not\leq_T H$. The latter property is known as *cone avoidance*.

In this talk, we provide a short survey on the known results about the nature of computation of Ramsey's theorem, and generalize them to a natural class of Ramsey-like theorems encompassing many statements studied in reverse mathematics. We show that this class admits a maximal statement satisfying cone avoidance and use it as a criterion to re-obtain many existing proofs of cone avoidance. This maximal statement asserts the existence, for every k-coloring of $[\omega]^n$, of an infinite subdomain $H \subseteq \omega$ over which the coloring depends only on the sparsity of its elements. This confirms the intuition that Ramsey-like theorems compute Turing degrees only through the sparsity of their solutions.

A Formal Analysis of Timing Channel Security via Bucketing

Tachio Terauchi
Waseda University
terauchi@waseda.jp

This paper investigates the effect of bucketing in security against timing channel attacks. Bucketing is a technique proposed to mitigate timing channel attacks by restricting a system's outputs to only occur at designated time intervals, and has the effect of reducing the possible timing channel observations to a small number of possibilities. However, there is little formal analysis on when and to what degree bucketing is effective against timing channel attacks. In this paper, we show that bucketing is in general insufficient to ensure security. Then, we present two conditions that can be used to ensure security of systems against adaptive timing channel attacks. The first is a general condition that ensures that the security of a system decreases only by a limited degree by allowing timing-channel observations, whereas he second condition ensures that the system would satisfy the first condition when bucketing is applied and hence becomes secure against timing-channel attacks. Further, we show that the bucketing technique can be applied compositionally in conjunction with the constant-time-implementation technique to increase their applicability. While we instantiate our contributions to timing channel and bucketing, many of the results are actually quite general and are applicable to any side channels and techniques that reduce the number of possible observations on the channel.

Contributed talks

Extracting the Fan Functional *

Ulrich Berger
Computer Science department
Swansea University

Consider a continuous functional $F : (\mathbf{N} \to \mathbf{B}) \to \mathbf{N}$ where $\mathbf{N}$ is the type of natural numbers and $\mathbf{B}$ is the type of Booleans, both endowed with the discrete topology, and the function space $\mathbf{N} \to \mathbf{B}$ carries the pointwise topology. By a compactness argument (Koenig's Lemma or Fan Theorem), F is uniformly continuous. Therefore, there exists a least modulus of uniform continuity for F, that is, a least natural number m such that F equates any two arguments that coincide below m. The mapping $F \mapsto m$ is called *Fan Functional*. Gandy showed that the Fan Functional is computable. Later it was shown that it is computable even in the restricted sense of Kleene's schemata (S1-S9), or, equivalently, PCF. We show that the Fan Functional is the computational content of a constructive proof of the uniform continuity of F. The proof takes place in a constructive theory with strictly positive inductive and coinductive definitions extended by an operator $!A$ whose realizability semantics signifies that A holds for trivial reasons and therefore is realized by *any* object of the right type. In our application this capures the fact that a continuous functional of type 2 does not inspects its argument beyond the point of continuity.

Regarding the constructive logical theory and its realizability interpretation this is joint work with Hideki Tsuiki.

*This work was supported by IRSES Nr. 612638 CORCON and Nr. 294962 COMPUTAL of the EC, the JSPS Core-to-Core Program, A. Advanced research Networks and JSPS KAKENHI 15K00015 as well as the Marie Curie RISE project CID (H2020-MSCA-RISE-2016-731143).

Second-order arithmetic, comprehension scheme and bar recursion

Valentin Blot
IRIF, CNRS, Université Paris-Diderot
LSV, Inria, CNRS, École Normale Supérieure Paris-Saclay

Abstract

In 1962, Spector extended Gödel's Dialectica interpretation of arithmetic with an operator of bar recursion providing computational content for the axiom of countable choice in classical logic. In 1971, Girard defined a very different, impredicative interpretation of second-order arithmetic using polymorphic lambda-calculus (system F). Even though the correspondence between the two logical systems is well known, no computational correspondence has been found so far.

We take a first step towards such a correspondence and define a computational interpretation of second-order arithmetic presented as an extension of first-order arithmetic with the axiom scheme of comprehension. We interpret the latter through bar recursion and obtain a simply-typed interpretation of second-order arithmetic.

In 1962, Spector defined bar recursion [6] to interpret the axiom of countable choice in the setting of Gödel's Dialectica interpretation [4]. Much later, Berardi, Bezem and Coquand defined a similar (but stronger) operator to interpret the axioms of countable and dependent choice [1] in the setting of Kreisel's modified realizability[5].

These interpretations provide computational content for proofs in second-order arithmetic through a combination of a well-known logical translation of second-order arithmetic into first-order arithmetic with countable choice and the interpretation of countable choice by bar recursion.

We make this combination computational and define a direct interpretation of second-order arithmetic by Berardi-Bezem-Coquand's version of bar recursion. We discuss the computational meaning of the bar recursive interpretation of the axiom scheme of comprehension that combines the backtracking interpretation of the excluded middle with the construction of a choice sequence by bar recursion.

As an application, we define a translation of Girard's polymorphic lambda calculus (system F [3]) into simply-typed lambda-calculus with bar recursion. For each term of system F there exists a proof of its termination in second-order arithmetic. Applying our interpretation to such a proof gives the translation in simply-typed lambda-calculus with bar recursion of the term of system F.

This presentation is based on previously published work [2].

References

[1] Stefano Berardi, Marc Bezem, and Thierry Coquand. On the Computational Content of the Axiom of Choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.

[2] Valentin Blot. An interpretation of system F through bar recursion. In *32nd ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2017.

[3] Jean-Yves Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *2nd Scandinavian Logic Symposium*, pages 63–69. North-Holland, 1971.

[4] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958.

[5] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957*, Studies in Logic and the Foundations of Mathematics, pages 101–128. North-Holland Publishing Company, 1959.

[6] Clifford Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, 1962.

Semantics, Logic, and Verification of *Exact Real Computation*

Franz Brauße[1], Pieter Collins[2], Johannes Kanig[3], SunYoung Kim[4], Michal Konečný[5], Gyesik Lee[6], Norbert Müller[1], Eike Neumann[5], Sewon Park[7], Norbert Preining[8], Martin Ziegler[7]
[1]Universität Trier, [2]Maastricht University, [3]AdaCore, [4]Yonsei University,
[5]Aston University, [6]Hankyong University, [7]KAIST, [8]Accelia Inc.

***Abstract*—This work formalizes *Exact Real Computation* (ERC): a paradigm combining (i) *algebraic* imperative programming of/over abstract data types (ADTs) for *continuous* structures with (ii) a selection and sound semantics of primitives computable in the sense of Recursive *Analysis*, that is, by means of approximations — yet presented to the user as exact.**

We specify a small imperative programming language for the ADT of real (i.e., including transcendental) numbers with rigorous semantics: arguments are provided, passed to and received from calls to functions (like e^x), and operated on *exactly* — with partial inequality predicate and multivalued binary `select` and continuous conditional (aka *parallel if*) operations — yet *realizing* a function (again like e^x) requires only to *approximate* its return value up to guaranteed absolute error 2^p for any given $p \in \mathbb{Z}$: closure under composition is implicit. We prove this language Turing-*complete*: a partial real function is computable in the sense of Recursive Analysis iff it can be expressed in ERC; and similarly for function*al*s.

Three simple numerical problems demonstrate both the convenience and novel control-flow considerations of this approach to Reliable Numerics: (I) multivalued integer rounding, (II) solving systems of linear equations, and (III) simple root finding. For rigorously specifying and arguing about such non-extensional computations, we propose a two-sorted structure over integers and reals, and prove its first-order theory both decidable and 'model complete': thus reflecting the elegance inherent to real (as opposed to rational/floating point) numbers. Rules of Hoare Logic are extended to support formal correctness proofs in ERC.

I. Motivation, Introduction, Overview

Based on Logic, the Theory of Computation provides fundamental concepts and tools devised to achieve and assert correctness, thus enabling modern modular software engineering — for problems over discrete structures: Common continuous realms (like real numbers) arising in Numerics arguably lack behind regarding rigorous treatment [1, p.412]. Best (?) practice commonly resorts to heuristics and 'recipes' [2] with vague specification [3, `e04bbc`], focussing on legacy encodings [4] which taint the elegance that made Mathematics move from rational to real numbers in the first place. Although often successful in practice, numerical codes may be flawed with sometimes dramatic consequences [5]–[7]. *Recursive Analysis* offers a sound algorithmic foundation to reliable computation on real numbers, functions, compact Euclidean subsets, and more general spaces [8], [9]:

> Call $x \in \mathbb{R}$ computable if some Turing machine can, for every $p \in \mathbb{Z}$, print the numerator $a_p \in \mathbb{Z}$ of dyadic rational $a_p 2^p$ approximating x up to error 2^p.

This notion has pleasant properties, such as closure under arithmetic as well as many transcendental functions [10, §4]. Moreover it leads to a Computational Complexity Theory [11] whose predictions [12] agree with the performance of practical implementations in reliable numerics [13]. Only, the underlying Turing machine model is inconvenient to code in [14]. The algebraic model [15], [16] aka realRAM or Blum-Shub-Smale Machine on the other hand is intuitive and prevalent in Computational Geometry, but neglects the influence of internal precision on the cost of operations, and its test for equality exhibits superrecursive power [17]. Indeed, "Do not test for equality!" is like the first commandment of Numerics whose rounding errors tend to taint mathematical equations. But which real comparisons are permitted, then? Strict inequality "$x > 0$" would allow to express equality via the Boolean combination "$\neg(x > 0) \wedge \neg(-x > 0)$".

The present work proposes and develops *Exact Real Computation* (ERC), a paradigm reconciling and combining the best of the two worlds: based on the algebraic model, but with additional multivalued primitives and a modified both sound and computable semantics of tests in the sense of Recursive Analysis.

Paradigm 1: ERC code realizing a real user function

f receives and operates on real arguments x *exactly* — yet with *partial* comparisons, and multivalued logical `select` and continuous conditional "$b\ ?\ x : y$" to achieve total correctness. It may even call some other real (user or predefined) functions g to receive and use their return values: again, *exactly*. However the value returned by the user's ERC code for f merely needs to *approximate* $f(x)$: up to guaranteed error 2^p for integer argument $p \in \mathbb{Z}$ given in addition to real x. ■

Note that this conception underlies, e.g., Newton's Method.

Overview: This is a short version of arXiv:1608.05787v3: We specify syntax and axiomatic semantics of a small imperative programming language for the ADT of real (i.e., including transcendental) numbers: with variables of two basic types — integer and real numbers — and with one-dimensional fixed-length arrays over each; with partial real comparison predicate, multivalued binary `select`, and continuous conditional (aka *parallel-if*); with WHILE loops and (real and multivalued integer) functions for subroutine calls. Programming in ERC regarding multivaluedness and using new/modified primitives is demonstrated with three numerical example problems: (I) multivalued integer rounding, (II) Gaussian Elimination for matrices of given rank, and (III) finding 1D simple roots. We prove soundness and adequacy, namely Turing-completeness: a real function is computable in the sense of Recursive Analysis iff it can be expressed in ERC; and we extend this to function*al*s. We then propose a two-sorted logical structure for rigorously specifying and arguing about the behaviour of such non-extensional programs; and show its first-order theory decidable and 'model-complete'. For verifying program correctness formally, we adapt and extend the classical Floyd-Hoare Logic to this structure.

Acknowledgements: We thank Andrej Bauer, Cyril Cohen, Jeehoon Kang, and Alex Simpson for seminal discussions, suggestions and feedback. This work was supported by the International Research & Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (grant NRF-2016K1A3A7A03950702), by the European Union's Horizon 2020 MSCA IRSES project 731143, and by the German Research Foundation (DFG) Grant MU 1801/5-1.

References

[1] P. Linz, "A critique of numerical analysis," *Bulletin of the American Mathematical Society*, vol. 19, no. 2, pp. 407–416, 1988.
[2] W. H. Press, *Numerical recipes: The art of scientific computing*. Cambridge university press, 2007.
[3] T. N. A. G. (NAG), "The nag library," Oxford, United Kingdom.
[4] D. Monniaux, "The pitfalls of verifying floating-point computations," *ACM Transactions on Programming Languages and Systems*, vol. 30, no. 3, May 2008.
[5] R. Skeel, "Roundoff error and the patriot missile," *SIAM News*, vol. 25, no. 4, p. 11, July 1992.
[6] J.-M. Jézéquel and B. Meyer, "Design by contract: The lessons of ariane," *Computer*, vol. 30, no. 1, pp. 129–130, January 1997.
[7] I. Holand, "The loss of the sleipner condeep platform," in *Proc. 1st International DIANA Conference on Computational Mechanics*, G. M. Kusters and M. A. Hendriks, Eds. Springer, Dordrecht, 1994.
[8] A. M. Turing, "On computable numbers, with an application to the "Entscheidungsproblem"," *Proceedings of the London Mathematical Society*, vol. 42, no. 2, pp. 230–265, 1937.
[9] M. B. Pour-El and J. I. Richards, *Computability in analysis and physics*. Cambridge Univ. Press, 2017.
[10] K. Weihrauch, *Computable Analysis*. Berlin: Springer, 2000.
[11] K.-I. Ko, *Complexity Theory of Real Functions*, ser. Progress in Theoretical Computer Science. Boston: Birkhäuser, 1991.
[12] J. M. Borwein and P. B. Borwein, *Pi and the AGM*. Wiley, 1987.
[13] Y. Kanada, "計算機による円周率計算（特集 円周率 π)," *Journal of Mathematical Culture*, vol. 1, no. 1, pp. 72–83, 2003.
[14] E. V. Arnold Schönhage, Andreas F. W. Grotefeld, *Fast algorithms*. B.I. Wissenschaftsverlag, 1994.
[15] B. Poizat, *Les petits cailloux*, ser. Nur al-mantiq wa-al-ma´rifah. Aléas, 1995.
[16] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation*. Springer, 1998. [Online]. Available: http://www.worldcat.org/oclc/37004484
[17] P. Boldi and S. Vigna, "Equality is a jump," *Theoretical computer science*, vol. 219, no. 1-2, pp. 49–64, 1999.
[18] S. A. Cook, "Soundness and completeness of an axiom system for program verification," *SIAM Journal on Computing*, vol. 7, no. 1, pp. 70–90, 1978.

A constructive Coq library for the mechanization of undecidability

Yannick Forster (Saarland University)
Dominique Larchey-Wendling (CNRS – LORIA)

January 15th, 2019

Abstract

We present a constructive library which contains both entry points and tools to mechanize undecidability results in Coq's type theory. Among the entry points are halting problems for Turing machines, binary stack machines, Minsky machines, decision problems like the Post correspondence problem, or entailments in intuitionistic linear logic. The tools we use to derive those results are reductions based on simulators, low level compilers, logical encodings of computational models or embeddings. We exploit the computability of all functions definable in constructive type theory and thus do not have to rely on a concrete model of computation, enabling the reduction proofs to focus on correctness properties.

The undecidability of low-level problems like the halting problem is usually shown by deriving a logical contradiction from the assumption that the problem is decidable. For even slightly more advanced problems, such a direct approach becomes infeasible and proofs are instead done by giving a (many-one) reduction from another undecidable problem to the problem to be shown undecidable. A proof by reduction in general amounts to defining the reduction function, showing that it is correct as a reduction and giving an argument for its computability. Most of these proofs rely on subtle details and have to be checked carefully. They could thus be a prime example for the use of interactive theorem provers to assist ongoing research. Mechanizations of undecidability proofs are vary rare in the literature. There are two main obstacles in our eyes: first, proofs on paper mostly omit the invariants needed for the full verification of the reduction and rather rely on an intuitive justification of the correctness of the reduction. Second, they omit the actual computability proof, which would amount to the full formal verification of a program in the chosen model of computation implementing the potentially complex reduction. Both these gaps have to be closed for the formalisation of reductions in a proof assistant.

Due to its commitment to effective methods, constructive type theory can only represent total computable functions and thus, in the following characterization of *decidability* of a problem $P : X \to \mathbb{P}$

$$\operatorname{dec} P := \exists f : X \to \{0,1\}, \forall x : X, P\, x \leftrightarrow f\, x = 1$$

the property of $f : X \to \{0,1\}$ being a total computable function is an unnecessary requirement. Within the dependently typed framework of (axiom free) Coq,

decidability can equivalently by characterized by the existence of a (dependent) decider of type:

$$\operatorname{dec} P := \forall x : X, \{P\,x\} + \{\neg P\,x\}$$

Moving on to the notion of *undecidability*, the classical approach of characterizing it as the logical negation of decidability $\neg(\operatorname{dec} P)$ would always lead to an unprovable statement because the (informative) excluded middle formula $\forall P : \mathbb{P}, \{P\} + \{\neg P\}$ is consistent with the type theory. Of course, postulating such a powerful axiom introduces non-computable functions in the logic.

Contrary to the above negative notion of classical undecidability $\neg(\operatorname{dec} P)$, we propose a positive approach to constructive undecidability, by defining it inductively through two rules:

$$\frac{}{\operatorname{undec} \mathsf{Halt}} \qquad\qquad \frac{\operatorname{dec} Q \to \operatorname{dec} P \qquad \operatorname{undec} P}{\operatorname{undec} Q}$$

the first rule stating that some seed Halt (e.g. the halting problem for Turing machines) is undecidable, the second rule stating that undecidability is closed under the Turing reduction $\operatorname{dec} Q \to \operatorname{dec} P$ (written $P \preceq_T Q$) which maps a decider for Q into a decider for P. This definition nicely corresponds to the usual practice in papers establishing undecidability results, where a problem is almost always related to a previously proved undecidable problem via some reduction. In practice, the full power of Turing reductions is unnecessary and reductions occur the sub-class of many-one reductions:

$$P \preceq Q := \exists f : X \to Y, \forall x : X,\ P\,x \leftrightarrow Q(f\,x)$$

where again, as we work in constructive type theory, the classical requirement of computability on f can be dropped from the definition of $P \preceq Q$. Using this approach, we get a potentially incomplete but *safe* definition of the notion of undecidability which is already enough to mechanize a wide range of results.

Indeed, our library already contains the following computational problems: single tape Turing machines termination problems, string rewrite systems related problems, Post correspondence problems (PCP) [1], binary stack machines termination (with just PUSH and POP instructions), Minsky (or register) machines (MM) termination (with just INC and DEC instructions), and undecidability results on logical entailment up to that of intuitionistic linear logic [3, 4]. External contributions to our framework already include a reduction from call-by-value λ-calculus to Halt [5] and a reduction from PCP to the first order logic problems of validity, provability and satisfiability [2]. The latest additions to our library include reductions from and to μ-recursive algorithm termination [6] and a reduction from MM to diophantine equations which, combined with the previous reductions, provides a fully mechanized constructive proof of the negative answer to Hilbert's 10th problem famously closed by I. Matiassevitch in the early 70's. The proposed talk is intended to summarize our results recently published at CPP'19 [4] and present some recent additions including the undecidability of satisfiability for diophantine equations.

References

[1] Yannick Forster, Edith Heiter, and Gert Smolka. Verification of pcp-related computational reductions in coq. In *International Conference on Interactive Theorem Proving*, pages 253–269. Springer, 2018.

[2] Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in coq, with an application to the entscheidungsproblem. In Mahboubi and Myreen [7], pages 38–51.

[3] Yannick Forster and Dominique Larchey-Wendling. Towards a library of formalised undecidable problems in coq: The undecidability of intuitionistic linear logic. *Workshop on Syntax and Semantics of Low-level Languages, Oxford*, 2018.

[4] Yannick Forster and Dominique Larchey-Wendling. Certified undecidability of intuitionistic linear logic via binary stack machines and minsky machines. In Mahboubi and Myreen [7], pages 104–117.

[5] Yannick Forster and Gert Smolka. Weak call-by-value lambda calculus as a model of computation in Coq. In *Interactive Theorem Proving - 8th International Conference, ITP*, pages 189–206. Springer, 2017.

[6] Dominique Larchey-Wendling. Typing total recursive functions in coq. In *Interactive Theorem Proving - 8th International Conference, ITP*, pages 371–388. Springer, 2017.

[7] Assia Mahboubi and Magnus O. Myreen, editors. *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*. ACM, 2019.

Tiered complexity at higher order

Emmanuel Hainry, Bruce Kapron, Jean-Yves Marion and Romain Péchoux

Extended abstract

In [1], Kapron and Steinberg introduce restrictions on Oracle Turing Machines to characterize the class of basic feasible functionals (BFF).

The Oracle Turing Machines they consider M_ϕ are Turing Machines with one query tape for oracle calls. If a query is written on such a tape and the machine enters a query-state, then the machine outputs the oracle's answer on the query tape in one step.

Definition 1. *Given an OTM M_ϕ and an input $\boldsymbol{a}$, let $m_{\boldsymbol{a}}^{M_\phi}$ be the maximum of the size of the input $\boldsymbol{a}$ and of the biggest oracle's answer in the run of machine on input $\boldsymbol{a}$ with oracle ϕ. A machine M_ϕ has:*

- *a polynomial step count if there is a polynomial P such that for any input $\boldsymbol{a}$ and oracle ϕ, M runs in time bounded by $P(m_{\boldsymbol{a}}^{M_\phi})$.*
- *a finite length revision if there exists a natural number n such that for any oracle and any input, in the run of the machine, the number of times it happens that an oracle answer is bigger than the input and all of the previous oracle answers is at most n.*
- *a finite lookahead revision if there exists a natural number n such that for any oracle and any input, in the run of the machine, it happens at most n times that a query is posed whose size exceeds the size of all previous queries.*

Definition 2 (Strong and Moderate Poly-Time).

- SPT *is the class of second order functions computable by an OTM with a polynomial step count and finite length revision.*
- MPT *is the class of second order functions computable by an OTM with a polynomial step count and finite lookahead revision.*

For a given class of functionals X, let $\lambda(X)$ be the set of simply typed lambda-terms where a constant symbol is available for each element of X. Let $\lambda(X)_2$ be the set of type two functionals represented by type two terms of $\lambda(X)$.

They obtain the following characterization of BFF:

Theorem 1. $\lambda(\mathrm{MPT})_2 = \lambda(\mathrm{SPT})_2 = BFF$

The main interest of this characterization is that it does not require the use of polynomial at order 2 and the semantics restrictions on the OTM are very natural.

We are now interested in finding a static analysis criterion allowing to ensure that a program computes a function of BFF. Our target is an implicit computational complexity characterization of BFF based on Kapron and Steinberg's restrictions.

For that purpose, we consider a simple imperative programming language with basic operators and oracles:

$$\begin{array}{lll} \texttt{Expressions} & \texttt{e}, \texttt{e}_1, \ldots, \texttt{e}_n & ::= \texttt{x} \mid \texttt{op}(\texttt{e}_1, \ldots, \texttt{e}_{ar(\texttt{op})}) \mid \phi(\texttt{e}_1 \upharpoonright \texttt{e}_2) \\ \texttt{Commands} & \texttt{c}, \texttt{c}_1, \texttt{c}_2 & ::= \texttt{skip} \mid \texttt{x} := \texttt{e} \mid \texttt{c}_1 \ ; \ \texttt{c}_2 \\ & & \quad \mid \texttt{if(e)\{c}_1\texttt{\} else \{c}_2\texttt{\}} \mid \texttt{while(e)\{c\}} \\ \texttt{Programs} & \texttt{p}_\phi & ::= \texttt{c return x} \end{array}$$

We introduce a type system with k tiers (a tier can be viewed as a natural number) inspired by the type system of [2] that prevents data flows from lower tiers to higher tiers. Types are triplet of tiers.

Let $[\![\mathrm{ST}]\!]$ be the set of functions computed by terminating and typable programs.

The main properties ensured by the type system are:

- a standard non-interference property ensuring that computations on higher tiers do not depend on lower tiers.
- a polynomial step count.
- a finite lookahead revision property.

and consequently, we have a soundness property:

Proposition 1 ($[\![\mathrm{ST}]\!] \subseteq$ MPT). *If* $\texttt{p}_\phi \in \mathrm{ST}$*, then it computes a second order function over words in* MPT*.*

We also have a stability by lambda-closure property:

Proposition 2. $\lambda([\![\mathrm{ST}]\!])_2 = [\![\mathrm{ST}]\!]$

contrarily to MPT and SPT that are strictly embedded in BFF.

However, completeness is still an open issue (the answer is likely to be negative for general oracles). For showing completeness, we need to be able to write a while loop guarded by a time counter containing the polynomial step count. Any polynomial P can be encoded by a program $\texttt{p}_\phi \in \mathrm{ST}$ (on unary data) but it is unclear whether $m_{\mathbf{a}}^{M_\phi}$ can be computed in this class. Is the function $\mathbf{a}, \phi \mapsto m_{\mathbf{a}}^{M_\phi}$ in $[\![\mathrm{ST}]\!]$? If the answer is negative, the study should be focused on oracles with the "good" properties. We have at least a static and decidable method (type inference can be done in polynomial time) for certifying programs to compute functions in BFF.

.

References

1. Bruce M. Kapron and Florian Steinberg. Type-two polynomial-time and restricted lookahead. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 579–588, 2018.
2. Jean-Yves Marion. A type system for complexity flow analysis. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 123–132, 2011.

Left-Normal Translation

Akira Hashida[1] and Nao Hirokawa[2]

[1] Graduate School of Advanced Science and Technology, JAIST, 1-1 Asahidai, Nomi, Japan, s1710158@jaist.ac.jp
[2] School of Information Science, JAIST, 1-1 Asahidai, Nomi, Japan, hirokawa@jaist.ac.jp

How to compute normal forms is a fundamental question in term rewriting. Consider the orthogonal constructor rewrite system:

$$\begin{array}{lll} 1\colon & \mathsf{nth}(x:xs,\mathsf{0}) \to x & 3\colon \quad \mathsf{from}(x) \to x:\mathsf{from}(\mathsf{s}(x)) \\ 2\colon & \mathsf{nth}(x:xs,\mathsf{s}(y)) \to \mathsf{nth}(xs,y) & 4\colon \quad \mathsf{0}+x \to x \end{array}$$

The term $\mathsf{nth}(\mathsf{from}(\mathsf{0}),\mathsf{0}+\mathsf{s}(\mathsf{0}))$ can be computed as follows:

$$\begin{aligned} \mathsf{nth}(\underline{\mathsf{from}(\mathsf{0})},\mathsf{0}+\mathsf{s}(\mathsf{0})) &\to \mathsf{nth}(\mathsf{0}:\mathsf{from}(\mathsf{s}(\mathsf{0})),\underline{\mathsf{0}+\mathsf{s}(\mathsf{0})}) \\ &\to \underline{\mathsf{nth}(\mathsf{0}:\mathsf{from}(\mathsf{s}(\mathsf{0})),\mathsf{s}(\mathsf{0}))} \to \mathsf{nth}(\underline{\mathsf{from}(\mathsf{s}(\mathsf{0}))},\mathsf{0}) \\ &\to \underline{\mathsf{nth}(\mathsf{s}(\mathsf{0}):\mathsf{from}(\mathsf{s}(\mathsf{0})),\mathsf{0})} \to \mathsf{s}(\mathsf{0}) \end{aligned}$$

Here underlines indicate rewrite positions. Due to the presence of the non-terminating subterm $\mathsf{from}(\mathsf{0})$, not all rewrite sequences reach the normal form. For instance, the leftmost outermost strategy, which rewrites subterms at leftmost outermost rewrite positions, yields the infinite sequence:

$$\begin{aligned} \mathsf{nth}(\underline{\mathsf{from}(\mathsf{0})},\mathsf{0}+\mathsf{s}(\mathsf{0})) &\to \mathsf{nth}(\mathsf{0}:\underline{\mathsf{from}(\mathsf{s}(\mathsf{0}))},\mathsf{0}+\mathsf{s}(\mathsf{0})) \\ &\to \mathsf{nth}(\mathsf{0}:(\mathsf{s}(\mathsf{0}):\underline{\mathsf{from}(\mathsf{s}(\mathsf{s}(\mathsf{0})))}),\mathsf{0}+\mathsf{s}(\mathsf{0})) \to \cdots \end{aligned}$$

So the leftmost outermost strategy is not a *normalizing* strategy for orthogonal systems in general.

The purpose of this note is to illustrate how the leftmost outermost strategy can be used for normalizing terms. Our technique is based on two landmark results in rewriting. One is the Normalization Theorem by O'Donnell [1]. The theorem says that the leftmost outermost strategy is normalizing for *left-normal* orthogonal systems. Left-normality means that in the left-hand side of each rule, functions symbol precede all variables in prefix notation. While this strategy is easy to implement, left-normality is a severe restriction. In fact, the first and second rewrite rules violate the restriction as $\mathsf{0}$ and s are behind x.

Another is Huet and Lévy's result on the *needed strategy* [2]. They showed that it is a normalizing strategy for *all* orthogonal systems. This notable generality comes with the price that it is actually an uncomputable strategy. That is why the notion of *strong sequentiality* [3] was introduced. For strongly sequential orthogonal systems *needed* positions (i.e. rewrite positions in the needed strategy) are computable. The above orthogonal system is strongly sequential,

and the first rewrite sequence ending with $\mathsf{s}(0)$ is obtained by the needed strategy. Needed positions can be computed in linear time, while its algorithm is nontrivial.

We present a translation technique, dubbed *left-normal translation*. Recall non-left-normal rules 1 and 2 of our running example. Introducing a fresh function symbol f, we translate them into left-normal forms:

$$\begin{array}{lll} 0'\colon & \mathsf{nth}(x : xs, y) \to \mathsf{f}(y, x, xs) & 3\colon \quad \mathsf{from}(x) \to x : \mathsf{from}(\mathsf{s}(x)) \\ 1'\colon & \mathsf{f}(0, x, xs) \to x & 4\colon \quad 0 + x \to x \\ 2'\colon & \mathsf{f}(\mathsf{s}(y), x, xs) \to \mathsf{nth}(xs, y) & \end{array}$$

Now the system is left-normal, and therefore the Normalization Theorem applies. Moreover, the leftmost outermost strategy simulates the needed strategy in the original system:

$$\begin{array}{ll} \mathsf{nth}(\underline{\mathsf{from}(0)}, 0 + \mathsf{s}(0)) & \to \underline{\mathsf{nth}(0 : \mathsf{from}(\mathsf{s}(0)), 0 + \mathsf{s}(0))} \\ \to \mathsf{f}(\underline{0 + \mathsf{s}(0)}, 0, \mathsf{from}(\mathsf{s}(0))) & \to \underline{\mathsf{f}(\mathsf{s}(0), 0, \mathsf{from}(\mathsf{s}(0)))} \\ \to \mathsf{nth}(\underline{\mathsf{from}(\mathsf{s}(0))}, 0) & \to \underline{\mathsf{nth}(\mathsf{s}(0) : \mathsf{from}(\mathsf{s}(\mathsf{s}(0))), 0)} \\ \to \underline{\mathsf{f}(0, \mathsf{s}(0), \mathsf{from}(\mathsf{s}(0)))} & \to \mathsf{s}(0) \end{array}$$

The idea behind the translation is to shift potential needed positions (known as indexes [3]) in rules to leftmost outermost positions. At the MLA 2019 workshop we will show that the left-normal translation is applicable to all strongly sequential, orthogonal constructor systems.

References

1. M.J. O'Donnell. Computing in Systems Described by Equations, volume 58 of *LNCS*, Springer, 1977.
2. G. Huet and J. J. Lévy. Computations in Orthogonal Term Rewriting Systems, I. In *Computational Logic — Essays in Honor of Alan Robinson*, pages 395–414. The MIT Press, 1991.
3. G. Huet and J. J. Lévy. Computations in Orthogonal Term Rewriting Systems, II. In *Computational Logic — Essays in Honor of Alan Robinson*, pages 415–443. The MIT Press, 1991.

Proof nets for first-order additive linear logic

Willem Heijltjes Dominic Hughes Lutz Straßburger

January 15, 2019

In this talk we will present canonical proof nets for first-order additive linear logic, the fragment of linear logic with sum, product, and first-order universal and existential quantification. The central challenge is to combine the witnessing information to existential quantifiers with the behaviour of the additive conjunction. The latter creates multiple "slices", each containing a different version of the same quantifier, with a potentially different witness.

The challenge is met by upending the traditional evaluation of an existential quantifier by an immediate substitution: instead, the substitution is recorded separately, at each axiom link in the proof net. The result is a canonical notion of proof nets for this logic. The main thrust of the work resolves the technical consequences of the design. Efficient and intuitive correctness and sequentialization are given by "coalescence", an additive version of multiplicative contractibility; in essence, this is top-down sequentialization by simple graph rewriting. A main contribution is an intricate geometric correctness condition, which subtly combines "slicing" correctness for propositional additives with "dependency" correctness for quantification. Cut-elimination involves the composition of the witnessing substitutions in two proof nets by "composition + hiding" in the style of game semantics.

A further contribution is the observation — following recent work by Dominic Hughes for first-order multiplicative linear logic — that witnessing information can be omitted from first-order additive proof nets altogether, and reconstructed via unification. This yields a further, coarser notion of proof net that factors out any inessential choice in witness assignment.

Details can be found in the technical report [1].

References

[1] Willem B. Heijltjes, Dominic J. D. Hughes, and Lutz Straßburger. *Proof nets for first-order additive linear logic.* Inria Research Report RR-9201, 2018

On the logical structure of certain bar induction and choice principles

Hugo Herbelin[1] and Nuria Brede[2]

[1] IRIF, CNRS, University Paris Diderot, Inria, France
hugo.herbelin@inria.fr
[2] University of Potsdam, Germany.
nuria.brede@uni-potsdam.de

There is a large body of literature on bar induction principles, on the Fan Theorem, as well as on on their contrapositives, the Axiom of Dependent Choice and König's Lemma, as well as on the relations between them. Our purpose here is to analyse a number of different formulations of these principles in a systematic way, highlighting dualities, analysing their computational contents, studying the classical, intuitionistic and linear logic equivalences, as well as integrating the Ultrafilter Theorem into the picture.

Our classification is based on a duality between choice principles (axiom of Dependent Choice, König's Lemma, Ultrafilter Theorem) and bar principles (Bar Induction, Fan Theorem). While the choice principles are thought of as connecting different definitions of existence of infinite paths in trees, the bar principles are considered as connecting different definitions of well-foundedness of trees. With this idea in mind, we use the definitions below, which all apply to a predicate T over the set A^* of finite sequences of elements of a given domain A. Our focus being purely logical, we do not impose any arithmetical restriction (such as decidability) on the predicate.

We use the letter a to range over elements of A, the letter u to range over the elements of A^*, n to range over the natural numbers $\mathbb{N}$ and α to range over functions from $\mathbb{N}$ to A.

Equivalent concepts on dual predicates	
T is a tree $\forall u \forall a\,(u \star a \in T \to u \in T)$	T is monotone $\forall u \forall a\,(u \in T \to u \star a \in T)$
T is progressing $\forall u\,(u \in T \to \exists a\, u \star a \in T)$	T is hereditary $\forall u\,((\forall a\, u \star a \in T) \to u \in T)$

Dual concepts on dual predicates	
infinite-branch-style	*well-foundedness-style*
Intensional concepts	
T has unbounded paths $\forall n \exists u\,(\lvert u\rvert = n \wedge \forall v\,(v \leq u \to v \in T))$	T is uniformly barred $\exists n \forall u\,(\lvert u\rvert = n \to \exists v\,(v \leq u \wedge v \in T))$
T is staged infinite[1] $\forall n \exists u\,(\lvert u\rvert = n \wedge u \in T)$	T is staged barred[1] $\exists n \forall u\,(\lvert u\rvert = n \to u \in T)$
T is a spread $\langle\rangle \in T \wedge T$ progressing	T is resisting[1] T hereditary $\to \langle\rangle \in T$
pruning of T $\nu X.\lambda u.(u \in T \wedge \exists a\, u \star a \in X)$	adherence[1] of T $\mu X.\lambda u.(u \in T \vee \forall a\, u \star a \in X)$
T is leaking[1] or productive[1] $\langle\rangle \in$ pruning of T	T is inductively barred $\langle\rangle \in$ adherence of T
Extensional concepts	
T has an infinite branch $\exists \alpha \forall u\,(u$ initial segment of $\alpha \to u \in T)$	T is barred $\forall \alpha \exists u\,(u$ initial segment of $\alpha \wedge u \in T)$

For each of the above definitions, we may adopt a classical (i.e. interpreting $\exists$ classically), intuitionistic (i.e. interpreting $\exists$ intuitionistically), or linear reading (i.e. additionally interpreting $\to$ as a linear arrow and conjunction either as a tensor or as a cartesian product). The binders μ and ν define respectively a smallest and greatest fixpoint.

[1] Not being aware of an established terminology for this concept, we use here our own terminology.

We take the definition of inductively barred, leaking, having an infinite branch and barred as references and define:

- Bar Induction (BI_A): $\forall T \in \mathcal{P}(A^*)$ (T barred $\to$ T inductively barred)
- Dependent Choice (DC_A): $\forall T \in \mathcal{P}(A^*)$ (T leaking $\to$ T has an infinite branch)

Standard formulations of Dependent Choice, e.g. $\forall T \in \mathcal{P}(A^*)$ (T spread $\to$ T has an infinite branch) can be shown linearly equivalent to the above definition. Standard formulations of the Fan Theorem, e.g. $\forall T \in \mathcal{P}(A^*)$ (T barred $\to$ T uniformly barred), for A finite, can be shown intuitionistically equivalent to BI_A. Standard formulations of König's Lemma, e.g. $\forall T \in \mathcal{P}(A^*)$ (T tree $\wedge$ T staged infinite $\to$ T has an infinite branch), for A finite and T decidable, can be shown equivalent to DC_A, up to LLPO. In particular, we rely on the following equivalences between these notions:

- "Leaking" is equivalent to the "existence of a subset which is a spread".
- "Inductively barred" is equivalent to "all supersets are resisting".
- On a finite domain and for a decidable predicate, the implication of "leaking" from "having unbounded paths" is equivalent to LLPO, following [1].

We did not define the property "u initial segment of α" yet. On a binary domain $\mathbb{B} \triangleq \{0, 1\}$, α can be represented either as a function from $\mathbb{N}$ to $\mathbb{B}$, or as a functional relation over $\mathbb{N} \times \mathbb{B}$, or as a predicate over $\mathbb{N}$. For instance, in the latter case, "u initial segment of α" would be defined by the following clauses:

$$\frac{}{\langle\rangle \text{ initial segment of } \alpha} \qquad \frac{u \text{ initial segment of } \alpha \qquad |u| \in \alpha}{u \star 0 \text{ initial segment of } \alpha} \qquad \frac{u \text{ initial segment of } \alpha \qquad |u| \notin \alpha}{u \star 1 \text{ initial segment of } \alpha}$$

These three possible different representations of a two-valued function lead to three different forms of the Fan Theorem on $\mathbb{B}$ of increasing strength, where the version for $\alpha \in \mathbb{N} \to \mathbb{B}$ is the strongest and the definition for $\alpha \in \mathcal{P}(\mathbb{N})$ the weakest, with some classical reasoning needed to go from $\alpha \in \mathcal{P}(\mathbb{N})$ to α functional relation on $\mathbb{N} \times \mathbb{B}$, and the axiom of unique choice needed to go next to $\alpha \in \mathbb{N} \to \mathbb{B}$. In particular, the weakest version is provable in second-order intuitionistic arithmetic, in the same way as Weak König's Lemma is provable from ACA_0 when formulated using $\alpha \in \mathcal{P}(\mathbb{N})$ [2].

We will also investigate a generalisation of the Weak Fan Theorem, i.e. of the Fan Theorem on $\mathbb{B}$, to a dual of the Ultrafilter Theorem, as well as the dual generalisation of Weak König's Lemma to the Ultrafilter Theorem. Again, different formulations are possible depending on how an ultrafilter is represented. In particular, the different formulations of the Weak Fan Theorem will arise as special cases of the corresponding formulations of the Ultrafilter Theorem on a countable domain.

References

[1] Hajime Ishihara. Constructive Reverse Mathematics: Compactness Properties. *From Sets and Types to Topology and Analysis: Towards Practicable Foundations for Constructive Mathematics*, 48, 10 2005.

[2] Stephen G. Simpson. *Subsystems of Second Order Arithmetic.* Perspectives in Logic. Cambridge University Press, 2nd edition, 2009.

Presenting de Groot duality of stably compact spaces

Tatsuji Kawai

Japan Advanced Institute of Science and Technology
tatsuji.kawai@jaist.ac.jp

Goubault-Larrecq [2] showed that the de Groot duality of stably compact spaces induces a family of dualities on various powerdomain constructions. For example, he showed that the dual of the Smyth powerdomain of a stably compact space X is the Hoare powerdomain of the dual X^{d}; the dual of the Plotkin powerdomain of X is the Plotkin powerdomain of X^{d}; and the same holds for the probabilistic powerdomain.

In this talk, we give a constructive account of this phenomenon in the setting of strong proximity lattice, a point-free representation of a stably compact space due to Jung and Sünderhauf [3]. To this end, we introduce a notion of *continuous entailment relation* [4], which can be thought of as a presentation of a strong proximity lattice by generators and relations. The notion is a variant of an entailment relation with the interpolation property due to Coquand and Zhang [1]. Here, the structure due to Coquand and Zhang is strengthened so that it has an intrinsic duality which reflects the de Groot duality of stably compact spaces. With a suitable notion of morphism, the category of continuous entailment relations becomes equivalent to that of strong proximity lattices. This allows us to reason about various constructions on strong proximity lattices using generators and relations.

In particular, the notion of continuous entailment relation allows us to identify de Groot duals of stably compact spaces presented by generators and relations by looking at the duals of their presentations. With this observation, we reproduce the results of Goubault-Larrecq [2] in a purely point-free and constructive setting. The examples include lower, upper, and Vietoris powerlocales; patch topology; and the space of probabilistic valuations. These examples illustrate simplicity of our approach by which we can reason about the de Groot duality of stably compact spaces.

References

1. T. Coquand and G.-Q. Zhang. A representation of stably compact spaces, and patch topology. *Theoret. Comput. Sci.*, 305(1-3):77–84, 2003.
2. J. Goubault-Larrecq. De Groot duality and models of choice: angels, demons and nature. *Math. Structures Comput. Sci.*, 20(2):169–237, 2010.
3. A. Jung and P. Sünderhauf. On the duality of compact vs. open. *Ann. N. Y. Acad. Sci.*, 806(1):214–230, 1996.
4. T. Kawai. Presenting de Groot duality of stably compact spaces. *ArXiv e-prints*, 1812.06480.

On randomized polynomial-time approximability of real numbers and sets

Akitoshi Kawamura Ulysse Léchine

In its simplest form, complexity theory often deals with decision problems. A *decision problem* A is specified by a set $\operatorname{dom} A \subseteq \{0,1\}^*$ of allowable inputs and, for each $u \in \operatorname{dom} A$, the correct answer $A(u) \in \{0,1\}$. When $\operatorname{dom} A = \{0,1\}^*$, we say that A is *total*[1] and write $A \in \mathsf{total}$. An algorithm (or a Turing machine, to be formal) for A is required to work correctly for all inputs in $\operatorname{dom} A$. Specifically, we write P (or BPP, respectively) for the class of decision problems solved by a deterministic (or randomized, respectively) polynomial-time algorithm in the sense that, given any input $u \in \operatorname{dom} A$, it outputs $A(u)$ (with probability $\geq 2/3$, respectively)[2]. These complexity classes are generally considered to match what can be done efficiently in the real world. Whether $\mathsf{BPP} = \mathsf{P}$, i.e., whether randomness increases the power of polynomial-time computation, is an important open question in complexity theory.

In the field of computable analysis, the theory of computability and complexity is applied to problems involving real numbers [1, 3]. In an early paper in this line of research, Chou and Ko [2] proposed two notions of efficiently computable subsets of $[0;1]^d$ (the d-dimensional unit cube). A set $S \subseteq [0;1]^d$ is P-*recognizable* if there is an algorithm that, given a point $x \in [0;1]^d$ (appropriately represented by an oracle) and $n \in \mathbf{N}$, determines in time polynomial in n whether x belongs to S, except when x is within distance 2^{-n} of the boundary of S. A set S is P-*approximable* if there is an algorithm that similarly determines whether x belongs to S, except for x in a set of measure 2^{-n}. When S is a region enclosed by a simple rectifiable curve, it can be shown that P-recognizability implies P-approximability. The converse claim is related to the questions of BPP versus P:

[1] Some authors say "problems" to mean total problems, and instead call problems in our sense *promise problems*, because the algorithm is promised that the input belongs to $\operatorname{dom} A$.

[2] There are other classes defined by randomized algorithms, depending on different notions of "solving" a problem (i.e., how the algorithm is allowed to err). For BPP, it is not hard to see that the specific bound of $2/3$ does not matter: it can be replaced by any constant in the open interval $(1/2;1)$ without changing the class BPP.

Theorem 1 ([2, Theorem 3.9])**.** *Let $d \geq 2$. In the following,* (1) $\Longrightarrow$ (2) $\Longrightarrow$ (3).

(1) $\mathsf{BPP} = \mathsf{P}$.

(2) *Let $S \subseteq [0;1]^d$. If S is* P*-approximable, then S is* P*-recognizable.*

(3) $\mathsf{BPP} \cap \mathsf{total} = \mathsf{P} \cap \mathsf{total}$.

Note that (3) does not easily imply (1). For all we know, a problem A in BPP may not be a restriction of a problem in $\mathsf{BPP} \cap \mathsf{total}$: possibly A is solved only by an algorithm that, for some inputs outside $\operatorname{dom} A$, fails to give any answer with probability $\geq 2/3$. Proving or disproving this implication would be a big leap forward in complexity theory. We proved a weaker version of it in Theorem 2 below.

What happens if we weaken the above condition (2) by restricting the class of sets S? Chou and Ko [2, Theorem 3.10] considered the restriction to (d-dimensional) rectangles, and essentially proved the implication (2) $\Longrightarrow$ (3) of Theorem 2 below, suggesting a relation to tally problems. A decision problem A is *tally* ($A \in \mathsf{tally}$) if $\operatorname{dom} A \subseteq \{0\}^*$; it is *tally total* ($A \in \mathsf{tally\text{-}total}$) if $\operatorname{dom} A = \{0\}^*$. We show that, surprisingly, the three conditions are equivalent in this analogue of Theorem 1:

Theorem 2. *Let $d \geq 1$. The following are equivalent.*

(1) $\mathsf{BPP} \cap \mathsf{tally} = \mathsf{P} \cap \mathsf{tally}$.

(2) *Let S be a rectangle in $[0;1]^d$. If S is* P*-approximable, then S is* P*-recognizable.*

(3) $\mathsf{BPP} \cap \mathsf{tally\text{-}total} = \mathsf{P} \cap \mathsf{tally\text{-}total}$.

We can also show that if the analogue of Theorem 2 were to be proved with very sparse problems instead of tally problems (A is *very sparse* if $\operatorname{dom} A \cap \{0,1\}^n$ contains at most one element for each $n \in \mathbf{N}$), then the part (3) $\Longrightarrow$ (1) of Theorem 1 would follow. In that regard, tally problems seem an interesting subcase to study.

References

[1] V. Brattka, P. Hertling, and K. Weihrauch. A tutorial on computable analysis. In S. B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms: Changing Conceptions of What is Computable*, pages 425–491. Springer, 2008.

[2] A. Chou and K. Ko. Computational complexity of two-dimensional regions. *SIAM Journal on Computing*, 24:923–947, 1995.

[3] K. Ko. *Complexity Theory of Real Functions*. Birkhäuser Boston, 1991.

Exact Bounds in Classical Natural Deduction

Delia Kesner
IRIF (CNRS and Université Paris-Diderot)
firstname.name@irif.fr

Pierre Vial
Inria (LS2N and IMT)
firstname.name@inria.ff

January 15, 2019

In this paper, we use non-idempotent typing (also known as *quantitative* typing) to obtain an exact measure, for any typable $\lambda\mu$-term t, of the number of reduction steps necessary to normalize t and of the size of the normal form of t. This statement requires some clarifications, that we are going to give now.

- *Lambda-mu calculus, classical logic and control operators:* via the Curry-Howard correspondence, the λ-calculus is a computational interpretation of *intuitionistic* natural deduction. The $\lambda\mu$ is an extension of the λ-calculus, which a computational interpretation of *classical* natural deduction. Concretely, classical features enables *control operators*, handling *continuations* and *backtracking* (*i.e.* the possibility to come back in the execution of a program). The $\lambda\mu$-calculus was introduced by Parigot in 92 [7], two years after Griffin proved that the control operator `callcc` from `Scheme` is typable with Peirce Law $(((A \to B) \to A) \to A)$, which is known to give classical logic.
- *Non-idempotent intersection types and quantitativity:* non-idempotent intersection types, introduced by Gardner [4] and de Carvalho [3] have been extensively used to obtain *upper bounds* for the number of reduction steps and normal forms. In the wake of Girard's Linear Logic [5], they forbid duplication for types, so that $A \wedge A \neq A$ (non-idempotency), where $\wedge$ is the intersection operator. Then, $t : A$ means that t may be used *once* as a term of type A and $t : A \wedge A$ that it may be used *twice* and so on. Non-idempotent intersection has been used to characterize various notions of normalization (head, weak, strong, weak head, linear head) while providing quantitative information, *e.g.* "t normalizes in less than 42 reduction steps".
- *Exact measures:* Bernadet and Lengrand [2] used quantitative type to statically characterize the exact length of maximal reduction sequence in the case of strong normalization. Building on this work, Accattoli, Lengrand and Kesner [1] developed a general framework encapturing exact lengths of reduction sequences along with the exact sizes of normal forms for the head, leftmost-outermost and maximal strategies[1]. In this framework, a typing judgment has the form $\Gamma \vdash^{(\ell,f)} t : \tau$, where, under an assumption called *tightness*, ℓ gives the number of reduction steps from t to its normal form t' and f is the size of t'.

Normalization and reduction strategies in the $\lambda\mu$-calculus

Furthering the work of Accattoli-Lengrand-Kesner [1], we use non-idempotent intersection and *union* types that we introduced in [6] to obtain exact measures for reduction strategies and normal form. Concretely, we present a unified type system which is parametrized so that it characterizes head, weak and strong normalization. This system relies on the distinction arrow types typing the abstractions of a redex, which are going to be fired (including the created redexes), and the arrow types assigned to variables which have a residual in the normal form. The former are denoted with $\to$ (named *dynamic* arrows) whereas the latter are denoted with $\twoheadrightarrow$ (named *static* arrows). For instance, in the λ-calculus, a simple version (without intersection) of our system would give:

$$\Phi_1 = \dfrac{\dfrac{\dfrac{}{x : \bullet \vdash^{(0,1)} x : \bullet}\,(\mathsf{ax})}{\emptyset \vdash^{(1,0)} (\lambda x.x) : \bullet \to \bullet}\,(\mathsf{abs}) \qquad y : \bullet \vdash^{(0,1)} y : \bullet}{y : \bullet \vdash^{(1,1)} (\lambda x.x)y}\,(\mathsf{app})$$

[1] Maximal strategy are deterministic reduction strategies which yield a reduction sequence of maximal length for strongly normalizing terms and an infinite reduction sequence for non strongly normalizing terms

$$\Phi_2 = \dfrac{\dfrac{\dfrac{}{x : \bullet \rightharpoonup \bullet \vdash^{(0,1)} x : \bullet \rightharpoonup \bullet}\,(\mathsf{ax}) \qquad \Psi_u \triangleright \emptyset \vdash^{(0,f)} u : \bullet}{x : \bullet \rightharpoonup \bullet \vdash^{(0,2+f)} x\,u : \bullet}\,(\mathtt{app})}{\vdash^{(0,3+f)} \lambda x.x\,u : \bullet}\,(\mathtt{tight})$$

where Ψ_u types the normal form u whose size is f. In the Φ_1, the abstraction typing the identity $\lambda x.x$ has dynamic arrow, because the redex $(\lambda x.x)y$ is fired to obtain the normal form y whereas, in Φ_2:

- The application $x\,u$ is not a redex and x is not to be replaced with an abstraction (the subject $\lambda x.x\,u$ is a normal form). We may then type x with a static arrow type $\bullet \rightharpoonup \bullet$.
- The abstraction is not fired either, so that it is typed with a specific rule (`tight`) which does not conclude with an arrow. The fact that (`tight`) does not produce an arrow type indeed prevents the abstraction to be applied.

For any normalizing term t, the system gives the number of β-reduction steps (intuitionistic steps), the number of μ-reduction steps (classical steps) and the size of its normal form. Derivations conclude with judgment of the form $\Gamma \vdash^{(\ell,m,f)} t : \mathcal{U} \mid \Delta$, where t is a $\lambda\mu$-term, $\mathcal{U}$ a type and Γ and Δ are contexts (with Δ pertaining to classical features of the calculus). Under a tightness assumption subsuming that of [1], ℓ, m and f give the three expected values. This gives our main theorem:

Theorem. There is a type system $\mathcal{X}_S$ for the $\lambda\mu$-calculus, parametrized with $S \in \{\mathtt{hd}, \mathtt{lo}, \mathtt{mx}\}$ and based on non-idempotent intersection and union types, coming along with a condition on judgment called tightness such that:

- t is head normalizing iff it is $\mathcal{X}_{\mathtt{hd}}$-typable.
  In that case, there is actually a derivable tight judgment $\Gamma \vdash^{(\ell,m,f)} t : \mathcal{U} \mid \Delta$ such that ℓ is the number of β-steps and m is the number of μ-steps in the *head* reduction strategy, and f is the size of the *head* normal form of t.
- t is weakly normalizing iff it is $\mathcal{X}_{\mathtt{lo}}$-typable.
  In that case, there is actually a derivable tight judgment $\Gamma \vdash^{(\ell,m,f)} t : \mathcal{U} \mid \Delta$ such that ℓ is the number of β-steps and m is the number of μ-steps in the *leftmost-outermost* strategy and f is the size of the normal form of t.
- t is strongly normalizing iff it is $\mathcal{X}_{\mathtt{mx}}$-typable.
  In that case, there is actually a derivable tight judgment $\Gamma \vdash^{(\ell,m,f)} t : \mathcal{U} \mid \Delta$ such that ℓ is the number of β-steps and m is the number of μ-steps[2] in the *maximal* reduction strategy.

We will present a formalism which factorizes most rules, proofs and statements, including that of the above theorem.

References

[1] B. Accattoli, S. Graham-Lengrand, and D. Kesner. Tight typings and split bounds. *PACMPL*, 2(ICFP):94:1–94:30, 2018.

[2] A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.

[3] D. de Carvalho. *Sémantique de la logique linéaire et temps de calcul.* PhD thesis, Université Aix-Marseille, Nov. 2007.

[4] P. Gardner. Discovering needed reductions using type theory. In *TACS, Sendai*, 1994.

[5] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[6] D. Kesner and P. Vial. Types as Resources for Classical Natural Deduction. In *FSCD, September 4-7, Oxford, England*, pages 1–15, 2017.

[7] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *LPAR*, pages 190–201, 1992.

[2] In this case, f is the size of the normal form of t plus the cumulated size of the normal forms that are erased during reduction.

Entailment Checking Procedure for Symbolic Heap using Complete Cyclic Proof System

Daisuke Kimura[1], Makoto Tatsuta[2], and[3] Koji Nakazawa

[1] Toho University kmr@is.sci.toho-u.ac.jp
[2] National Institute of Informatics / Sokendai tatsuta@nii.ac.jp
[3] Nagoya University knak@i.nagoya-u.ac.jp

Abstract. This talk presents an entailment checker for a symbolic-heap system with user-defined inductively defined predicates. Symbolic heap is a simple form of formulas in the separation logic [4] that is often used in the context of verifying pointer-manipulating programs [1, 2]. Decision procedure for the entailment checking problem, which asks validity of given entailments, is important issue for achieving automated program verifier based on Hoare logic. The procedure in this talk is based on the proof-search algorithm given in the authors' paper [5], which proposes a complete cyclic-proof system for entailments of the symbolic heap system with a class of general inductive predicates. The basic algorithm is inefficient because of an inference rule, called the split-rule, which splits the separating conjunctions on both sides of an entailment. The split-rule requires to explore all possible branches of the spliting cases in order to obtain completeness. For reducing this inefficiency, this talk also discusses an idea for finding a correct branch systematically.

References

1. J. Berdine, C. Calcagno, and P. W. O'Hearn, A Decidable Fragment of Separation Logic, In: Proceedings of FSTTCS 2004, *LNCS* 3328 (2004), 97–109.
2. J. Berdine, C. Calcagno, and P. W. O'Hearn, Symbolic Execution with Separation Logic, In: Proceedings of APLAS 2005, *LNCS* 3780 (2005), 52–68.
3. J. Brotherston and A. Simpson, Sequent calculi for induction and infinite descent, *Journal of Logic and Computation 21* (6) (2011) 1177–1216.
4. J.C. Reynolds, Separation Logic: A Logic for Shared Mutable Data Structures, In: *Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science (LICS2002)* (2002) 55–74.
5. M. Tatsuta, K. Nakazawa and D. Kimura, Completeness of Cyclic Proofs for Symbolic Heaps, https://arxiv.org/abs/1804.03938, 2018

This work is partially supported by JSPS Core-to-Core Program (A. Advanced Research Networks).

On Cut-Elimination Theorem in Cyclic-Proof Systems

Koji Nakazawa[1], Daisuke Kimura[2], Tachio Terauchi[3], Hiroshi Unno[4], and Kenji Saotome[1]

[1] Nagoya University, Japan, {knak,saotomekenji}@sqlab.jp
[2] Toho University, Japan, kmr@is.sci.toho-u.ac.jp
[3] Waseda University, Japan, terauchi@waseda.jp
[4] University of Tsukuba, Japan, uhiro@cs.tsukuba.ac.jp

Abstract. Cyclic-proof systems [1, 2] are logical systems in which the induction is represented as cyclic structure of proof trees. Cyclic-proof systems are suitable for automated proof search, and they have been actively studied for several logics such as the first-order logic, the linear logic, and the separation logic. However, fundamental properties of the cyclic-proof systems are not well-known. In this talk, we discuss on the cut-elimination property for the cyclic-proof systems. In particular, we prove that the cut-elimination theorem does not hold in the cyclic-proof system for the separation logic. We expect that the proof idea will be extended to other logics such as the first-order logic and the logic of bunched implications, that is, we conjecture the cut-elimination does not hold in the cyclic-proof systems for these logics. We also have another (relatively) positive conjecture that any proof with cuts can be translated to a proof with cuts restricted to only those against induction hypotheses. We call this property quasi-cut-elimination property.
This work is partially supported by JSPS Core-to-Core Program (A. Advanced Re- search Networks).

References

1. J. Brotherston, D. Distefano, and R. L. Petersen, Automated cyclic entailment proofs in separation logic, In: *Proceedings of CADE-23* (2011) 131–146.
2. J. Brotherston, N. Gorogiannis, and R. L. Petersen, A Generic Cyclic Theorem Prover, In: Proceedings of APLAS 2012, *LNCS* 7705 (2012) 350–367.

NETS AND REVERSE MATHEMATICS

SAM SANDERS

Abstract. Nets are generalisations of sequences involving *uncountable* index sets; this notion was introduced about a century ago by Moore and Smith. They also established the generalisation to nets of various basic theorems of analysis due to Bolzano-Weierstrass, Dini, Arzelà, and others. This paper deals with the Reverse Mathematics study of these and related theorems about nets. Perhaps surprisingly, over Kohlenbach's base theory of *higher-order* Reverse Mathematics, the Bolzano-Weierstrass theorem for nets and the unit interval implies the Heine-Borel theorem *for uncountable covers*. Hence, the former theorem is *extremely hard* to prove (in terms of the usual hierarchy of comprehension axioms), but also *unifies* the concepts of sequential and open-cover compactness. Similarly, Dini's theorem for nets is equivalent to the aforementioned Heine-Borel property. Finally, we show that approximating nets by sequences is hard, but that the notion of continuity associated to nets is equivalent to 'epsilon-delta' continuity *without* the Axiom of Choice.

Extended abstract

The move to more and and more abstract mathematics can be quite concrete and specific: E. H. Moore presented a framework called *General Analysis* at the 1908 ICM in Rome ([2]) that was to be a 'unifying abstract theory' for various parts of analysis. For instance, Moore's framework captures various limit notions in one abstract concept ([3]). This theory also included a generalisation of the concept of *sequence* beyond countable index sets, nowadays called *nets* or *Moore-Smith sequences*. These were first described in [4] and then formally introduced by Moore and Smith in [5]. They also established the generalisation to nets of various basic theorems due to Bolzano-Weierstrass, Dini, and Arzelà ([5, §8-9]). In this paper, we study theorems about nets in higher-order *Reverse Mathematics* ([1]).

We show that the Bolzano-Weierstrass theorem *for nets* implies both the sequential and *uncountable* open-cover compactness of the unit interval, i.e. a nice unification result. Open-cover compactness is captured by HBU from [6] and *extremely hard* to prove: in terms of the usual hierarchy of comprehension axioms, full second-order arithmetic is needed to prove HBU. We further study the following theorems generalised to nets: the monotone convergence theorem, Arzelà's theorem, the monotone convergence theorem for nets of functions, and Dini's theorem. In each case, we obtain HBU, or a weakening of the latter from [8] similar to *weak weak König's lemma*; we refer to [9, X.1] for the latter lemma.

The aforementioned results imply that basic results about nets are extremely hard to prove. It may therefore seem desirable (and in line with the coding practice of classical Reverse Mathematics) to replace or approximate the limit process

School of Mathematics, University of Leeds & Dept. of Mathematics, TU Darmstadt
E-mail address: sasander@me.com.

involving nets by a 'countable' limit process involving *sequences*, i.e. if a net converges to some limit, then there should be a *sequence* in the net that also converges to the same limit. We show in that (an highly elementary instance of) the latter 'sub-sequence property' implies the *Lindelöf lemma* for $\mathbb{R}$, which is at least[1] as hard to prove as HBU. Finally we establish the local equivalence between 'epsilon-delta' continuity and the notion of continuity provided by nets *without* using the Axiom of Choice. In other words, while basic properties of nets are hard to prove, nets can also obviate the need for extra axioms.

References

[1] Ulrich Kohlenbach, *Higher order reverse mathematics*, Reverse mathematics 2001, Lect. Notes Log., vol. 21, ASL, 2005, pp. 281–295.

[2] E. H. Moore, *On a Form of General Analysis with Aplication to Linear Differential and Integral Equations*, Atti IV Cong. Inter. Mat. (Roma,1908) **2** (1909), 98–114.

[3] ———, *Introduction to a Form of General Analysis*, Yale University Press, 1910.

[4] ———, *Definition of Limit in General Integral Analysis*, Proceedings of the National Academy of Sciences of the United States of America **1** (1915), no. 12, 628–632.

[5] E. H. Moore and H. L. Smith, *A General Theory of Limits*, Amer. J. Math. **44** (1922), no. 2, 102–121.

[6] Dag Normann and Sam Sanders, *On the mathematical and foundational significance of the uncountable*, Journal of Mathematical Logic, `https://doi.org/10.1142/S0219061319500016` (2018).

[7] ———, *Uniformity in mathematics*, Submitted, arXiv: `https://arxiv.org/abs/1808.09783` (2018).

[8] ———, *Representations in measure theory*, In preparation (2019).

[9] Stephen G. Simpson, *Subsystems of second order arithmetic*, 2nd ed., Perspectives in Logic, CUP, 2009.

[1]Note that LIN + WKL implies HBU by [9, IV.1], and there are versions of LIN that imply fragments of the Axiom of (coumtable) Choice (see [7, §5]), in contrast to HBU

Some formal proofs of isomorphy and discontinuity*

Florian Steinberg[1] and Holger Thies[2]

[1]INRIA, Saclay [2]Kyushu University, Fukuoka

In computable analysis a representation for a space X is a partial surjective mapping from the Baire space $\mathbb{N}^\mathbb{N}$ to X, i.e., a function $\delta\colon\subseteq \mathbb{N}^\mathbb{N} \to X$. A pair $\mathbf{X} = (X, \delta_\mathbf{X})$ of a space and its representation is called a represented space and an element φ of Baire space is called name of $x \in \mathbf{X}$ if $\delta(\varphi) = x$. Through a representation the notions of computability and continuity of operators on Baire space can be transfered to any represented space $\mathbf{X}$. Popular topics in computable analysis are proving mathematical problems computable or, if this is impossible, classifying their degree of incomputability [BG11, BDBP12, PS18]. A problem that often appears in such classifications is closed choice, where the task is "given a non-empty closed set $A \in \mathcal{A}(\mathbf{X})$ select an element $a \in A$". Here, a closed subset of a represented space is given by specifying positive information about its complement. Thus, for most choices of $\mathbf{X}$, this task is uncomputable and even discontinuous.

More formally, $\mathcal{A}(\mathbf{X})$ is defined by use of the space of open subsets $\mathcal{O}(\mathbf{X})$, which in turn (following for instance [Pau16]) can abstractly be described as the space of continuous functions from $\mathbf{X}$ to the Sierpiski space $\mathbb{S}$. Here, $\mathbb{S}$ has the two point set $\{\bot, \top\}$ as underlying set and the total function $\delta_\mathbb{S}$ specified by

$$\delta_\mathbb{S}(\varphi) = \top \quad \iff \quad \exists n \in \mathbb{N}\ \varphi(n) \neq 0$$

as representation. The function space construction from computable analysis [Wei00, Definition 3.3.13] provides a representation $[\delta_\mathbf{X} \to \delta_\mathbb{S}]$ of the continuous functions from $\mathbf{X}$ to $\mathbb{S}$ and we call the resulting represented space $\mathbb{S}^\mathbf{X}$. Next, identify a subset U of $\mathbf{X}$ with its characteristic function

$$\chi_U : \mathbf{X} \to \mathbb{S}, \quad \chi_U(x) := \begin{cases} \top & \text{if } x \in U, \\ \bot & \text{otherwise.} \end{cases}$$

Conveniently, χ_U is continuous if and only if U is open and therefore $\mathcal{O}(\mathbf{X})$ can be identified with $\mathbb{S}^\mathbf{X}$. Finally, $\mathcal{A}(\mathbf{X})$ is represented as the complements of opens, i.e., following the above

$$\delta_{\mathcal{A}(\mathbf{X})}(\varphi) = A \quad \iff \quad [\delta_\mathbf{X} \to \delta_\mathbb{S}](\varphi) = \chi_{X\setminus A}.$$

The task of closed choice on $\mathbf{X}$ is formalized as finding a realizer (in the sense of function realizabilty) of the multivalued function $C_\mathbf{X}\colon\subseteq \mathcal{A}(\mathbf{X}) \rightrightarrows \mathbf{X}$ defined by $C_\mathbf{X}(A) := A$. Or in words: a is an acceptable return value of $C_\mathbf{X}$ on input A if and only if a is an element of A. Note that this in particular means that the domain of $C_\mathbf{X}$ are the non-empty subsets of $\mathbf{X}$ and that a realizer can behave arbitrarily outside of the domain, i.e., no solution needs to be produced in this case.

The function space construction is quite complicated and for concrete spaces it is often possible to use simpler representations. For instance, one may make use of the fact that there exist infinite products and indeed $\prod_{n\in\mathbb{N}} \mathbf{X}$ is isomorphic to the function space $\mathbf{X}^\mathbb{N}$.

*This work was supported by JSPS KAKENHI Grant Number JP18J10407, by the Japan Society for the Promotion of Science (JSPS), Core-to-Core Program (A. Advanced Research Networks), by the ANR project FastRelax (ANR-14-CE25-0018-01) of the French National Agency for Research and by EU-MSCA-RISE project 731143 Computing with Infinite Data (CID).

Thus $\mathcal{O}(\mathbb{N}) \cong \prod_{i\in\mathbb{N}} \mathbb{S}$, where the right hand side uses the infinite tupling on Baire space as replacement for the more complicated function space construction. An even more concrete description of $\mathcal{O}(\mathbb{N})$ can be obtained by using the enumeration representation, where a name of an open set enumerates its elements. The representation of the corresponding concrete space $\mathbf{A}_\mathbb{N}$ of the closed subsets of the natural numbers is given by

$$\delta_{\mathbf{A}_\mathbb{N}}(\varphi) = \mathbb{N} \setminus \{n \in \mathbb{N} \mid \exists m \in \mathbb{N},\ \varphi(m) = n + 1\}.$$

That is, the information a name specifies about a closed set is an enumeration of its complement.

We formalized the proofs $\mathcal{A}(\mathbb{N}) \simeq \mathbf{A}_\mathbb{N}$ and also the isomorpy of the concrete and abstract spaces of open sets. Some of the steps in the proof are carried out in more generality than needed, for instance we prove $\mathbf{X}^\mathbb{N} \cong \prod_{i\in\mathbb{N}} \mathbf{X}$ and $\mathcal{O}(\mathbf{X}) \cong \mathcal{A}(\mathbf{X})$ (via taking the complement and realized by the identity) for an arbitrary represented spaces. The formalization is carried out in coqrep [Ste18], a coq library for computable analysis which is being developed by one of the authors. As an application we formalized a proof that closed choice on the naturals as function from $\mathbf{A}_\mathbb{N}$ to $\mathbb{N}$ is discontionuous and used the isomorphisms to conclude that the same is true for $C_\mathbb{N}$ in its original definition using $\mathcal{A}(\mathbb{N})$ as source space. Our proofs of isomorphy proceed by specifying a subset of the needed algorithms concretely, proving them correct and obtaining the remaining translations through compositionality. In particular we retain full executability even though some of our correctness proofs use classical reasoning for convenience. For obvious reasons the discontinuity statements are exempt from the executability claims.

The results themselves are well known and were picked for having fairly straight forward proofs while at the same time being well suited as examples for full formal proofs in the library that require some of the more advanced features it offers like the function space and infinite product constructions. During our work we also suplemented some previously missing parts to the library, for instance a full justification of the infinite product construction by means of a proof of the corresponding universal property. The whole project has meanwhile been made part of the coqrep library.

References

[BDBP12] Vasco Brattka, Matthew De Brecht, and Arno Pauly. Closed choice and a uniform low basis theorem. *Annals of Pure and Applied Logic*, 163(8):986–1008, 2012.

[BG11] Vasco Brattka and Guido Gherardi. Effective choice and boundedness principles in computable analysis. *Bulletin of Symbolic Logic*, 17(1):73–117, 2011.

[Pau16] Arno Pauly. On the topological aspects of the theory of represented spaces. *Computability*, 5(2):159–180, 2016.

[PS18] Arno Pauly and Florian Steinberg. Comparing representations for function spaces in computable analysis. *Theory of Computing Systems*, 62(3):557–582, 2018.

[Ste18] Florian Steinberg. coqrep. `https://github.com/FlorianSteinberg/coqrep`, 2018.

[Wei00] Klaus Weihrauch. *Computable Analysis*. Springer, Berlin/Heidelberg, 2000.

Completeness of Cyclic Proofs for Symbolic Heaps with Cone Inductive Definitions

Makoto Tatsuta
National Institute of Informatics / Sokendai, Japan
tatsuta@nii.ac.jp

Koji Nakazawa
Nagoya University, Japan
knak@i.nagoya-u.ac.jp

Daisuke Kimura
Toho University, Japan
kmr@is.sci.toho-u.ac.jp

Separation logic is successful for software verification in both theory and practice [1]. Decision procedure for symbolic heaps is one of the key issues. This paper proposes a cyclic proof system [2] for symbolic heaps with some class of inductive definitions, called cone inductive definitions [3], and shows its soundness and completeness. The decision procedure for entailments of symbolic heaps with cone inductive definitions is also given. The class of cone inductive definitions is useful since it contains skip lists, doubly linked lists, and lists of trees. Decidability for entailments of symbolic heaps with general form of inductive definitions is an important question. Completeness of cyclic proof systems is also an important question. The results of this paper answer both questions.

Acknowledgment

This is supported by Core-to-Core Program (A. Advanced Research Networks) of the Japan Society for the Promotion of Science.

References

[1] J. Berdine, C. Calcagno, and P. W. O'Hearn, Symbolic Execution with Separation Logic, In: Proceedings of APLAS 2005, *LNCS* 3780 (2005) 52–68.

[2] J. Brotherston, A Simpson, Sequent calculi for induction and infinite descent, *Journal of Logic and Computation 21* (6) (2011) 1177–1216.

[3] M. Tatsuta, K. Nakazawa, and D. Kimura, Completeness of Cyclic Proofs for Symbolic Heaps, arXiv:1804.03938, 2018.

Infinite Adequacy Theorem through Coinductive Definitions

Hideki Tsuiki,
Graduate School of Human and Environmental Studies,
Kyoto University, Kyoto, Japan

This is a joint work with Ulrich Berger.

We present adequacy theorem for an untyped language with full recursion and constructors for making pairs. In such a language, one can express infinite terms some of whose parts are $\bot$ (i.e., undefined). For example an infinite list that contains a $\bot$ can be represented. Our adequacy theorem is stated as follows.

> Suppose that M is a closed term whose meaning $[\![M]\!]$ is composed only with constructors and $\bot$. Then, as the computation of M proceeds, we have a sequence of constructor terms whose limit is the infinite term $[\![M]\!]$.

We are developing a logical system IFP (Infinite FixedPoint Logic), which is an extension of first-order logic with inductve and coinductive definitions. One can extract from a proof in IFP a program in this language. In programs, it may be the case that some part of the data cannot be specified and therefore should be left as $\bot$. For example, infinite Gray code is a representation of real numbers that takes advantage of this partiality [2]. With IFP, one can logically express such a specification and extract a program that produces a partially specified data. Therefore, in order for our program extraction system to be sound, it is required that a program really computes the data to the extent that is specified by the logic. This adequacy theorem shows this kind of soundness.

As a part of the proof, we used the adequacy theorem for finite terms given in [1], which requires a second-order argument that is beyond the power of IFP. However, we did other part of the proof in the way that can be formalized in IFP; we defined the required domain-theoretic notions and defined big-step and small-step reduction and proved their relations through induction and coinduction. In this sense, this can be seen as a working example of IFP proofs.

[1] U. Berger. Realisability for induction and coinduction with applications to constructive analysis. Jour. Universal Comput. Sci., 16(18):2535–2555, 2010.

[2] H. Tsuiki. Real Number Computation through Gray Code Embedding. Theoretical Computer Science, 284(2):467–485, 2002.

"First-order parts" of Weihrauch degrees

Keita Yokoyama

Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan,
y-keita@jaist.ac.jp

The Weihracuch degree of a binary relation on Baire space measures the power of uniform computation of a problem defined on Baire space. In the recent studies of Weihrauch degrees, it is seen that its structure resembles the structure of second-order arithmetic in the sense of reverse mathematics. In this study, we will introduce the "first-order part" of a Weihrauch degree by focusing on numerical consequences, and try to measure the first-order strength of degrees. Then we see that the first-order parts of degrees of arithmetical problems form a hierarchy corresponding to Kirby-Paris hierarchy of first-order arithmetic, and those can be classified with their first-order strength. This is a joint work with Damir Dzhafarov and Reed Solomon.