Finite semantics of polymorphism, complexity and the power of type fixpoints

Lê Thành Dũng Nguyễn
LIPN, UMR CNRS 7030
Université Paris 13, Sorbonne Paris Cité
Villetaneuse, France
lethanhdung.nguyen@lipn.fr

Thomas Seiller
CNRS, LIPN – UMR 7030 U. Paris 13
Université Paris 13, Sorbonne Paris Cité
Villetaneuse, France
seiller@lipn.fr

Paolo Pistone
Wilhelm Schickard Institut
Universität Tübingen
Tübingen, Germany
paolo.pistone@uni-tuebingen.de

Lorenzo Tortora de Falco
Dipartimento di Matematica e Fisica
Università Roma Tre
Rome, Italy
tortora@uniroma3.it

Long version: `https://hal.archives-ouvertes.fr/hal-01979009v1`**, cited as [20].**

Polymorphism is a central topic in theoretical computer science since the sixties. A breakthrough in its logical understanding was its analysis by means of second order quantifiers, that is the introduction of System F at the beginning of the seventies. Some years later, this considerable success led Jean-Yves Girard to develop a denotational semantics for System F [8], to get a deeper understanding of its computational features. Indeed, the general goal of denotational semantics is to give a "mathematical" counterpart to syntactic devices such as proofs and programs, thus bringing to the fore their essential properties. Sometimes this eventually results in improvements of the syntax: Linear Logic itself [9] arose precisely from the denotational model introduced in [8].

But denotational semantics is not just a matter of increasing our understanding of programming languages, it also has direct algorithmic applications. Let us mention:

- in the simply-typed lambda calculus (STλ), the *semantic evaluation* technique for complexity bounds, see Terui's paper [22] and references therein;
- in STλ extended with a fixed-point combinator, the semantic approach to *higher-order model checking* (HOMC) advocated by Salvati and Walukiewicz [23, 24] (see also [1, 15]).

The following little-known theorem illustrates both kinds of applications. Indeed, it is an implicit complexity result and, at the same time an instance of the correspondence between Church encodings and automata that HOMC generalizes to infinite trees.

Theorem 1 (Hillebrand & Kanellakis [14])**.** *The languages decided by* STλ *terms from Church-encoded binary strings to Church booleans*[1] *are exactly the* regular *languages.*

The main idea behind this result is to build a deterministic finite automaton (DFA) computing the denotation of its input string. Crucially, this relies on the existence of a *finite semantics* for STλ – such as the category of finite sets – which will provide the states of the DFA. In general, this finiteness property, or finer cardinality bounds, are key to these applications.

This theorem also holds when replacing STλ by propositional linear logic, which also admits finite semantics. In fact, Terui's solution to the complexity of STλ normalization at fixed order [22] relies on

[1] That is, STλ terms of type $((A \to A) \to (A \to A) \to (A \to A)) \to (o \to o \to o)$, where o is a base type and A may be chosen depending on the language, but is independent of the input string.

Submitted to:
DICE-FOPARA 2019

such a semantics. As for HOMC, Grellois and Melliès have developed an approach relying on models of linear logic [12, 11].

However, this concerns only *monomorphic* type systems, for a simple reason: equality is definable on the type of System F natural numbers, so its denotation in any non-trivial model must be infinite.

1 Contributions

Polymorphism, linearity, and type fixpoints Actually, second-order quantification is not the only culprit here: one can also blame the *non-linearity* of the System F integers. What we show in this paper is that a semantics for a purely linear language with impredicative polymorphism can be finite:

Theorem 2. *Second-order Multiplicative-Additive Linear Logic (*MALL_2*) admits finite semantics.*

This mere existence already allows us to clarify the difference in expressive power between MALL_2 and $\mu\mathrm{MALL}$, i.e. MALL with type fixpoints [2]. While $\mu\mathrm{MALL}$ can be translated in second-order LL with exponentials, it was argued informally that a translation to MALL_2 could not exist [2, §2.3]; since $\mu\mathrm{MALL}$, which can encode infinite data types, does not admit non-trivial finite semantics, we can now provide a clear proof of this impossibility [20, Theorem III.1].

The power of type fixpoints also appears in implicit complexity, in Baillot's characterization of polynomial time [3] in second-order Elementary Linear Logic[2] (ELL_2) with recursive types:

Theorem 3 ([3])**.** *The predicates which can be expressed as a function* $!\mathtt{Str} \multimap !^2\mathtt{Bool}$ *in* $\mu\mathrm{ELL}_2$ *are exactly the* polynomial-time *predicates.*

Note also that further developments building upon this result [4, 5] all make use of type fixpoints. An open question was whether the same result holds without them; using Theorem 2, we show that we get instead a characterization of *regular languages*:

Theorem 4 ([20, Theorem III.4])**.** *The languages which can be decided by a function* $!\mathtt{Str} \multimap !^2\mathtt{Bool}$ *in* ELL_2 *are exactly the* regular *languages.*

Our proof is directly inspired by that of Theorem 1 by Hillebrand and Kanellakis, unsurprisingly since inputs are Church-encoded in both cases.

Witness-erasing semantics of polymorphism In models of MALL_2, the main obstacle to finiteness is the existential quantifier. Indeed, while existential-free MALL_2 formulae have finitely many cut-free proofs thanks to linearity, existential variables may have witnesses of arbitrary size. To prove Theorem 2, the goal is therefore to "erase" these witnesses in the interpretation, in order to compress the proof to some bounded data depending only on the proven formula. Following the programming language point of view on existential types as abstract data types, this will mean remembering just enough information to determine their interaction with the generic (universally typed) programs which might use them.

Typically, all the proofs

$$\dfrac{\dfrac{\cdots}{\vdash A}}{\vdash \exists X.X}\,\exists$$

should be collapsed – recall that $\exists X.X$ is the impredicative encoding of the additive unit $\top$.

Our first implementation of this idea is purely syntactic: it consists in taking an *observational quotient* of the syntax. Although the model thus obtained is indeed finite [20, Proposition II.16] and has the

[2] In fact, Baillot works with Elementary *Affine* Logic, but unlike type fixpoints, weakening makes no difference as to complexity. See also Laurent's notes "Polynomial Time in Untyped Elementary Linear Logic".

advantages of simplicity and canonicity, we will see that it is not effective in several ways [20, Theorems II.17 and II.18], which may impede applications. Still, we believe that the finiteness proof for this quotient conveys important intuitions.

To get an effective model, we turn to Girard's semantics of polymorphism in *coherence spaces* [8, 9]. It is more concrete than its formulation using category-theoretic machinery (normal functors) could suggest, and as it turns out, Girard's definitions already lead to finite and computable denotations of MALL_2 types [20, Theorems IV.16 and IV.18]. We prove this by singling out a notion of *finite degree* which is preserved by MALL_2 connectives and ensures finiteness, and sketch a combinatorial presentation. By the way, the witness-erasing character of this model – indeed, the non-trivial computational contents of its existential introduction, which subsumes the cut rule – was already noted in [9, p. 57] as being "key to a semantic approach to computation".

Sub-polynomial implicit complexity Although we do not detail an application of the effectiveness of coherence spaces here, we lay the groundwork [20, Proposition IV.19] for a sequel paper [19] which will apply it to a conjectural implicit characterization of *deterministic logarithmic space* with a few distinctive features. As we shall discuss in the present paper, trying to go beyond regular languages in ELL_2 naturally leads to changing the representation of inputs, and this conjecture arises directly from pursuing this line of thought by taking inspiration from by Hillebrand's PhD thesis [13].

By performing semantic evaluation in coherence spaces, the sequel manages to establish a sub-polynomial upper bound – though not yet deterministic logspace soundness. We believe this approach to implicit complexity to be rather novel, as will be explained in [19]: unlike in previous works on characterizing logarithmic space via substructural logics [21, 7, 18], it seems that soundness cannot be established by variants of the Geometry of Interaction interpretation of the multiplicative-exponential fragment of linear logic. Let us stress, then, that this novelty is only made possible thanks to the semantic investigations undertaken here.

Let us also mention that our result on regular languages in ELL_2 betrays the original spirit of light logics [10] which consisted in bounding the complexity of normalization "geometrically", independently of types. Here, while geometry still plays an important structuring role, our fine-grained analysis requires to take into account the influence of types through semantics.

Finite models and parametricity We finally compare the two models presented with two properties arising from parametricity [17], a well-known approach to polymorphism. Parametricity is a desirable property from our viewpoint as it provides a "smallness" condition on the interpretation of quantifiers. In particular, *dinaturality*, proposed as a categorical formalisation of parametricity since [6], provides a third example of witness-erasing semantics, since the interpretation of quantifers as *ends/coends* leads to identify proofs with different witnesses. However, the coherent model is not dinatural [20, Proposition V.6] and the construction of finite parametric models of MALL_2 seems a non-trivial task, also as a consequence of [20, Proposition V.9], which shows that compact closed models do not satisfy the *constancy* property, another property of parametric models introduced in [16].

References

[1] Klaus Aehlig (2007): *A Finite Semantics of Simply-Typed Lambda Terms for Infinite Runs of Automata*. *Logical Methods in Computer Science* 3(3), doi:10.2168/LMCS-3(3:1)2007.

[2] David Baelde (2012): *Least and Greatest Fixed Points in Linear Logic*. *ACM Transactions on Computational Logic* 13(1), pp. 1–44, doi:10.1145/2071368.2071370.

[3] Patrick Baillot (2015): *On the expressivity of elementary linear logic: Characterizing Ptime and an exponential time hierarchy*. *Information and Computation* 241, pp. 3–31, doi:10.1016/j.ic.2014.10.005.

[4] Patrick Baillot, Erika De Benedetti & Simona Ronchi Della Rocca (2018): *Characterizing polynomial and exponential complexity classes in elementary lambda-calculus*. *Information and Computation* 261, pp. 55–77, doi:10.1016/j.ic.2018.05.005.

[5] Patrick Baillot & Alexis Ghyselen (2018): *Combining Linear Logic and Size Types for Implicit Complexity*. In: *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, pp. 9:1–9:21, doi:10.4230/LIPIcs.CSL.2018.9.

[6] E. S. Bainbridge, P. J. Freyd, A. Scedrov & P. J. Scott (1990): *Functorial Polymorphism*. *Theor. Comput. Sci.* 70(1), pp. 35–64, doi:10.1016/0304-3975(90)90151-7. Available at `http://dx.doi.org/10.1016/0304-3975(90)90151-7`.

[7] Ugo Dal Lago & Ulrich Schöpp (2016): *Computation by interaction for space-bounded functional programming*. *Information and Computation* 248, pp. 150–194, doi:10.1016/j.ic.2015.04.006.

[8] Jean-Yves Girard (1986): *The system F of variable types, fifteen years later*. *Theoretical Computer Science* 45, pp. 159–192, doi:10.1016/0304-3975(86)90044-7.

[9] Jean-Yves Girard (1987): *Linear logic*. *Theoretical Computer Science* 50(1), pp. 1–101, doi:10.1016/0304-3975(87)90045-4.

[10] Jean-Yves Girard (1998): *Light Linear Logic*. *Information and Computation* 143(2), pp. 175–204, doi:10.1006/inco.1998.2700.

[11] Charles Grellois (2016): *Semantics of linear logic and higher-order model-checking*. Ph.D. thesis, Univeristé Denis Diderot Paris 7. Available at `https://tel.archives-ouvertes.fr/tel-01311150/`.

[12] Charles Grellois & Paul-André Melliès (2015): *Relational Semantics of Linear Logic and Higher-order Model Checking*. In: *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, pp. 260–276, doi:10.4230/LIPIcs.CSL.2015.260.

[13] Gerd G. Hillebrand (1994): *Finite Model Theory in the Simply Typed Lambda Calculus*. Ph.D. thesis, Brown University, Providence, RI, USA.

[14] Gerd G. Hillebrand & Paris C. Kanellakis (1996): *On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi*. In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, IEEE Computer Society, pp. 253–263, doi:10.1109/LICS.1996.561337.

[15] Martin Hofmann & Jeremy Ledent (2017): *A cartesian-closed category for higher-order model checking*. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, Reykjavik, Iceland, pp. 1–12, doi:10.1109/LICS.2017.8005120.

[16] Giuseppe Longo, Kathleen Milsted & Sergei Soloviev (1993): *The Genericity Theorem and Parametricity in the Polymorphic Lambda-calculus*. *Theor. Comput. Sci.* 121(1-2), pp. 323–349, doi:10.1016/0304-3975(93)90093-9. Available at `http://dx.doi.org/10.1016/0304-3975(93)90093-9`.

[17] QingMing Ma & John C. Reynolds (1992): *Types, abstraction, and parametric polymorphism. II*. In: *Mathematical foundations of programming semantics (Pittsburgh, PA, 1991)*, *Lecture Notes in Comput. Sci.* 598, Springer, Berlin, pp. 1–40, doi:10.1007/3-540-55511-0_1.

[18] Damiano Mazza (2015): *Simple Parsimonious Types and Logarithmic Space*. In: *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, pp. 24–40, doi:10.4230/LIPIcs.CSL.2015.24.

[19] Lê Thành Dũng Nguyễn & Pierre Pradic (2019): *From normal functors to logarithmic space queries*. Available at `https://hal.archives-ouvertes.fr/hal-02024152`. Submitted.

[20] Lê Thành Dũng Nguyễn, Thomas Seiller, Paolo Pistone & Lorenzo Tortora De Falco (2019): *Finite semantics of polymorphism, complexity and the power of type fixpoints*. Available at `https://hal.archives-ouvertes.fr/hal-01979009`. Submitted.

[21] Ulrich Schöpp (2007): *Stratified Bounded Affine Logic for Logarithmic Space*. In: *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pp. 411–420, doi:10.1109/LICS.2007.45.

[22] Kazushige Terui (2012): *Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus*. In: *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, pp. 323–338, doi:10.4230/LIPIcs.RTA.2012.323.

[23] Igor Walukiewicz & Sylvain Salvati (2015): *Using models to model-check recursive schemes*. *Logical Methods in Computer Science* Volume 11, Issue 2.

[24] Igor Walukiewicz & Sylvain Salvati (2017): *Typing weak MSOL properties*. *Logical Methods in Computer Science* Volume 13, Issue 1.