Implicit automata in typed λ-calculi

Cécilia Pradic
Department of Computer Science, University of Oxford, United Kingdom

This is joint work with Lê Thành Dũng Nguyễn (LIPN, Paris Nord).

The research that we present here started out by exploring connections between the languages (resp. functions) recognized by automata (resp. transducers, i.e., automata with output) and those definable by programs in certain typed λ-calculi. While we are relating logic and automata, much like programming language theory and proof theory are linked via the Curry–Howard correspondence, our work does not fit in the "logics as specification languages" paradigm, exemplified by the equivalence of finite-state automata and Monadic Second-Order Logic (MSO). One could sum up the difference by analogy with the two main approaches to machine-free complexity: *implicit computational complexity (ICC)* and *descriptive complexity*. Both aim to characterize complexity classes without reference to a machine model, but the methods of ICC have a more computational flavor.

programming paradigm	declarative	functional
complexity classes	Descriptive Complexity	Implicit Computational Complexity
automata theory	subsystems of MSO	**our work**

To our knowledge, very few works had previously looked at this kind of "type-theoretic" or "proof-theoretic" ICC for automata. Let us mention a few recent papers concerning transducers [5, 2] and multi-head automata [19, 10]. Most importantly, our starting point is a remarkable result dating back to 1996:

▶ **Theorem 1** (Hillebrand & Kanellakis [8, Theorem 3.4]). *A language $L \subseteq \Sigma^*$ can be defined in the* simply typed λ-calculus *by some* closed λ-term *of type* $\texttt{Str}_\Sigma[A] \to \texttt{Bool}$ *for some type A (that may depend on L) if and only if it is a* regular language*.*

Let us explain this statement. We consider a grammar of simple types with a single base type: $A, B ::= o \mid A \to B$, and use the *Church encodings* of booleans $\texttt{Bool} = o \to o \to o$ and strings $\texttt{Str}_\Sigma = (o \to o) \to \ldots \to (o \to o) \to o \to o$ with $|\Sigma|$ arguments of type $(o \to o)$ where Σ is a finite alphabet. For types A and B, we write $B[A]$ for the substitution $B\{o := A\}$ of every occurrence of o in B by A.

Although little-known, Hillebrand and Kanellakis's theorem is not surprising in retrospect: strong connections between Church encodings and automata (see e.g. [18, 11]) have been exploited, in particular in *higher-order model checking*.

Linear and non-commutative types While there exist some results in implicit complexity based on the simply typed λ-calculus (e.g. [8]), many works in that area have taken inspiration from *linear logic* to design more sophisticated type systems, starting with two characterizations of polynomial time [7, 6]. We followed the same idea to characterize two transduction classes in Intuitionistic Linear Logic with additives (ILL). From now on, $\texttt{Str}_\Sigma$ denotes the linearized Church encoding $\texttt{Str}_\Sigma = (o \multimap o) \to \ldots \to (o \multimap o) \to o \to o$.

▶ **Theorem 2** ([14]). *A function $\Gamma^* \to \Sigma^*$ is* regular (see e.g. [12]) (resp. comparison-free polyregular [13]) *if and only if it can be defined by a closed term of type* $\texttt{Str}_\Gamma[A] \multimap \texttt{Str}_\Sigma$ (resp. $\texttt{Str}_\Gamma[A] \to \texttt{Str}_\Sigma$) *in ILL for some* purely linear *type A (that may depend on f).*

Here "purely linear" means that A does not contain any non-linear function arrow '$\to$'. We also provide a similar characterization of regular *tree* functions in [14]. What might be more surprising is our additional use of *non-commutativity* (a function must use its arguments in the same order that they are given in) to characterize a subclass of regular languages.

▶ **Theorem 3** ([15]). *A language $L \subseteq \Sigma^*$ is* star-free *if and only if it can be defined by a closed term of type* $\mathtt{Str}_\Sigma[A] \multimap \mathtt{Bool}$ *in an affine variant of* Intuitionistic Non-Commutative Linear Logic [17] *for some* purely linear *type A (that may depend on L).*

Denotational semantics meets categorical automata theory The key conceptual insight in our proof of Theorem 2 is to relate the semantics of purely linear ILL in monoidal closed categories with the representation of transducers in Colcombet and Petrişan's categorical framework for automata [3]. More precisely, we show that a variant of *copyless* (i.e. affine) *streaming string transducers* [12, §2] – a machine model for regular functions – can be formulated as $\mathcal{C}$-automata where $\mathcal{C}$ is a Dialectica-like completion of a category of string-valued registers, so $\mathcal{C}$ is monoidal closed for the same reason as Dialectica categories [4].

The notion of monoidal closure has a relevance for automata theory that goes beyond Theorem 2; intuitively, this is due to the important role that function spaces often play in automata constructions. To illustrate that, in [14], we gave abstract generalizations of the arguments showing that copyless SSTs may be determinized and that the composition of two regular functions may be implemented by a copyless SST, in terms of internal homsets. Interestingly, it is not clear to us if there is a nice condition analogous to monoidal closure over classes of *transition monoids* allowing to carry out those generalized arguments without introducing automata over monoidal closed categories.

Some automata-theoretic consequences We were thus led to define the aforementioned *comparison-free polyregular functions* by considering expressible functions in linear logic. This class of function is a natural restriction of polyregular functions [2], a class of string transductions whose outputs are of size at most polynomial in the output. We studied that class in [13] from the point of view of automata theory. While all arguments are rather unsurprising, the connection with λ-calculus helped us realize that the class was closed under composition, and provides an alternative proof of this fact leveraging the material of [14].

Furthermore, we also took inspiration from Theorem 3 and from the planar geometry of interaction (GoI) semantics [1] of non-commutative linear logic to design a new machine model for star-free languages and aperiodic regular functions (see [12, §3] for the latter): *planar two-way automata/transducers* [16]. It was previously known that two-way automata could be expressed as automata over a GoI category (a reformulation of the results of [9] in the framework of [3]), and that two-way transducers compute regular functions [12, §2].

References

1 Samson Abramsky. Temperley–Lieb Algebra: From Knot Theory to Logic and Computation via Quantum Mechanics. In Goong Chen, Louis Kauffman, and Samuel Lomonaco, editors, *Mathematics of Quantum Computation and Quantum Technology*, volume 20074453, pages 515–558. Chapman and Hall/CRC, September 2007. `doi:10.1201/9781584889007.ch15`.

2 Mikołaj Bojańczyk. Polyregular functions, 2018. `arXiv:1810.08760`.

3 Thomas Colcombet and Daniela Petrişan. Automata Minimization: a Functorial Approach. *Logical Methods in Computer Science*, 16(1), March 2020. `doi:10.23638/LMCS-16(1:32)2020`.

4 Valeria C. V. de Paiva. The Dialectica categories. In John W. Gray and Andre Scedrov, editors, *Contemporary Mathematics*, volume 92, pages 47–62. American Mathematical Society, Providence, Rhode Island, 1989. `doi:10.1090/conm/092/1003194`.

5 Henry DeYoung and Frank Pfenning. Substructural proofs as automata. In Atsushi Igarashi, editor, *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings*, volume 10017 of *Lecture Notes in Computer Science*, pages 3–22, 2016. `doi:10.1007/978-3-319-47958-3_1`.

6 Jean-Yves Girard. Light Linear Logic. *Information and Computation*, 143(2):175–204, June 1998. `doi:10.1006/inco.1998.2700`.

7 Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theoretical Computer Science*, 97(1):1–66, April 1992. `doi:10.1016/0304-3975(92)90386-T`.

8 Gerd G. Hillebrand and Paris C. Kanellakis. On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 253–263. IEEE Computer Society, 1996. `doi:10.1109/LICS.1996.561337`.

9 Peter Hines. A categorical framework for finite state machines. *Mathematical Structures in Computer Science*, 13(3):451–480, 2003. `doi:10.1017/S0960129503003931`.

10 Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic Proofs and Jumping Automata. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, volume 150 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. `doi:10.4230/LIPIcs.FSTTCS.2019.45`.

11 Paul-André Melliès. Higher-order parity automata. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, Reykjavik, Iceland, June 2017. IEEE. `doi:10.1109/LICS.2017.8005077`.

12 Anca Muscholl and Gabriele Puppis. The Many Facets of String Transducers. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. `doi:10.4230/LIPIcs.STACS.2019.2`.

13 Lê Thành Dũng Nguyễn, Camille Noûs, and Cécilia Pradic. Comparison-Free Polyregular Functions. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 139:1–139:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICALP.2021.139`.

14 Lê Thành Dũng Nguyễn. The planar geometry of first-order transductions, 2021. In preparation. Slides available at `https://nguyentito.eu/2021-01-links.pdf`.

15 Lê Thành Dũng Nguyễn, Camille Noûs, and Cécilia Pradic. Implicit automata in typed λ-calculi II: streaming transducers vs categorical semantics, 2020. `arXiv:2008.01050`.

16 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.135`.

17 Jeff Polakow and Frank Pfenning. Relating Natural Deduction and Sequent Calculus for Intuitionistic Non-Commutative Linear Logic. *Electronic Notes in Theoretical Computer Science*, 20:449–466, January 1999. `doi:10.1016/S1571-0661(04)80088-4`.

18 Sylvain Salvati. Recognizability in the simply typed lambda-calculus. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2009. `doi:10.1007/978-3-642-02261-6_5`.

19 Thomas Seiller. Interaction Graphs: Non-Deterministic Automata. *ACM Transactions on Computational Logic*, 19(3):21:1–21:24, August 2018. doi:10.1145/3226594.