Implicit Computational Complexity, Non-uniformity and the Lambda-Calculus

Damiano Mazza
CNRS, UMR 7030, LIPN, Université Paris 13, Sorbonne Paris Cité
Damiano.Mazza@lipn.univ-paris13.fr

Implicit computational complexity. Loosely speaking, the aim of implicit computational complexity is to replace clocks (or other explicit resource bounds) with certificates. For example, if we consider polynomial time computation, the idea is to define a structured programming language whose programs are guaranteed to be polytime by construction, *i.e.*, because they satisfy some syntactic condition, not because their execution is artificially stopped after a polynomial number of steps (which is the "explicit" definition used in computational complexity). At the same time, such a programming language must be expressive enough so that every polynomial time function may be somehow implemented. Notable early examples of such methodology are the work of Bellantoni and Cook [BC92], Leivant and Marion [LM93], and Jones [Jon99].

Although there is an annual international workshop on the topic,[1] implicit computational complexity is a niche research subject which rarely appears in mainstream programming languages conferences. Nevertheless, it is thematically relevant, both in terms of techniques (type systems, proof theory, semantics…) and potential applications (mostly program analysis [JHLM10, DLG11], but people like Patrick Baillot, Gilles Barthe and Ugo Dal Lago are starting to develop applications to the verification of cryptosystems).

Non-uniform computation. To keep the definition of cost model as simple and incontrovertible as possible, the "default" models used in computational complexity (Turing machines, RAMs…) are extremely low level. This has the effect of virtually erasing the fundamental structures present in data and algorithms: in Turing machines, there are no abstract data types, just strings; and all algorithms, albeit containing recursion, control structures, modules and such, become sets of tuples.

Although complexity theorists have been able to develop their research seemingly unimpeded by the above remark, in practice the theory has been most successful in proving non-trivial lower bounds (its *raison d'être*) when algorithms are expressed in computational models offering more structure than just sets of tuples or assembly-like instructions, such as decision trees, Boolean formulas and circuits (see for instance Part II of [AB09]). These models share the characteristic of being strongly finitary, in the sense that one program may only handle finitely many inputs (usually, all inputs of a given size). In order to compute a function in the usual sense (defined on data of arbitrary size), one has to consider infinite families of such programs (one for each input size). To stay within the realm of computable functions, one must impose a *uniformity* constraint, *i.e.*, that the members of the family are generated by a single "usual" program (*e.g.* a Turing machine, perhaps with bounded resources). There are well known connection theorems between, for instance, Turing machines and uniform families of circuits. However, the role of uniformity in practice is unclear: most lower bound proofs ignore it, *i.e.*, they apply to non-uniform families. It seems that, when it comes to bounded-resource computation, the gain in structure obtained by shifting from, say, Turing machines to Boolean circuits is valuable only at the local level. A lower bound proof for a circuit family exploits the local structure of the circuit processing data of size n; the fact that the family globally forms a coherent whole (an algorithm in the usual sense) is surprisingly unhelpful.

Non-uniformity in the λ-calculus. We propose to discuss the idea of non-uniformity from the point of view of functional programming and implicit computational complexity, following recent work by the author on an infinitary

[1] Developments in Implicit Computational Complexity (DICE), http://perso.ens-lyon.fr/patrick.baillot/DICE/.

affine λ-calculus [Maz12, Maz14].[2] The starting point is the "equation"

$$\frac{\text{affine } \lambda\text{-terms}}{\text{(infinitary affine) } \lambda\text{-terms}} = \frac{\text{Boolean circuits}}{\text{Turing machines (with advice)}}$$

where Turing machines with advice [AB09] are a non-uniform model essentially equivalent to non-uniform circuit families. The relationship between Boolean circuits and affine calculi was of course already known [Mai04, Ter04]. However, we are uncovering here a deeper, richer connection:

- the set $\ell\Lambda$ of affine λ-terms is endowed with a non-trivial topological structure, the Cauchy-completion of which yields an infinitary, but still affine calculus $\ell\Lambda_\infty$, in which the usual λ-calculus may be embedded. Furthermore, β-reduction, as a function on $\ell\Lambda_\infty$, is continuous, which means that reductions in $\ell\Lambda_\infty$ may be arbitrarily approximated by reductions in $\ell\Lambda$ (much like computation on real numbers may be approximated by computation on rational numbers, or arbitrary elements of an algebraic CPO may be approximated by compact elements).
- Terms representing data (Booleans, numerals, strings…) have the additional property of being topologically isolated. Combined with the above, this implies that, when $t\underline{w} \to^* \underline{b}$ is a computation deciding the membership of a word $w \in \{0,1\}^*$ to the language accepted by the term t, only a finite approximation t_m of t is sufficient to compute $\underline{b}$. If t is compared to a Turing machine M, then t_m is akin to the circuit approximating the behavior of M on inputs of size $|w|$. Using this fact, one may re-establish certain fundamental complexity-theoretic results, such as the Cook-Levin theorem (following [Lad75]), in a purely functional setting.
- In the above, t may also be arbitrarily non-uniform, *i.e.*, an infinite functional program whose behavior depends non-recursively on the input. The structure of infinite terms may be tamed using ideas of implicit computational complexity, obtaining a purely functional characterization of the non-uniform class P/poly (languages decided by arbitrary polynomial size circuits, uniform or not).
- An interesting perspective is to study the notion of uniformity via the infinitary affine λ-calculus. Here, uniformity may be defined in a purely algebraic way: the terms which are embeddings of usual λ-terms may be characterized by means of a partial equivalence relation. The hope is that, if successfully transported to circuits, this notion may help shedding some light on the role (or lack thereof) of uniformity in lower bound proofs.
- Another very interesting, but more speculative, line of research concerns the denotational semantics of a simply-typed version of $\ell\Lambda_\infty$. In the usual λ-calculus, semantic arguments have been used to show the impossibility of representing certain functions (*e.g.* subtraction) with simple types [BDS13]. Here, one may wonder whether such results translate into lower bound proofs for small circuit classes.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. CUP, 2009.

[BC92] Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

[BDS13] Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. CUP, 2013.

[DLG11] Ugo Dal Lago and Marco Gaboardi. Linear dependent types and relative completeness. *Log. Methods Comput. Sci.*, 8(4), 2011.

[JHLM10] S. Jost, K. Hammond, H.-W. Loidl, and M.Hofmann. Static determination of quantitative resource usage for higher-order programs. In *Proceedings of POPL*, pages 223–236, 2010.

[Jon99] Neil D. Jones. Logspace and ptime characterized by programming languages. *Theor. Comput. Sci.*, 228(1-2):151–174, 1999.

[Lad75] Richard E. Ladner. The circuit value problem is log-space complete for P. *SIGACT News*, 6(2):18–20, 1975.

[LM93] Daniel Leivant and Jean-Yves Marion. Lambda calculus characterizations of poly-time. *Fundam. Inform.*, 19(1/2), 1993.

[Mai04] Harry G. Mairson. Linear lambda calculus and ptime-completeness. *J. Funct. Program.*, 14(6):623–633, 2004.

[Maz12] Damiano Mazza. An infinitary affine lambda-calculus isomorphic to the full lambda-calculus. In *Proceedings of LICS*, pages 471–480, 2012.

[Maz14] Damiano Mazza. Non-uniform polytime computation in the infinitary affine lambda-calculus. In *Proceedings of ICALP*, pages 305–317, 2014.

[Ter04] Kazushige Terui. Proof nets and boolean circuits. In *Proceedings of LICS*, pages 182–191, 2004.

[2] A 4-page summary of [Maz14] is available on: `http://lipn.univ-paris13.fr/~mazza/papers/Ppoly-LCC.pdf`.