Non-uniform polynomial time via non-wellfounded parsimonious proofs

Matteo Acclavio  Department of Computer Science, University of Southern Denmark, macclavio@gmail.com
Gianluca Curzi  Department of Computer Science, University of Birmingham, g.curzi@bham.ac.uk
Giulio Guerrieri  LIS, Aix Marseille Université, giulio.guerrieri@lis-lab.fr

Linear logic (LL) [9] is a refinement of both classical and intuitionistic logic allowing control over computational resources. This is obtained by having a strong discipline on the use of resources in proofs thanks to the use of the exponential modality (denoted by !), which marks the distinction between those assumptions that can be used linearly, that is, exactly once, and those ones that are reusable at will. In the Curry-Howard interpretation, the exponential modality introduces non-linearity in functional programs: a proof of the linear implication !A ⊸ B is interpreted as a program returning an output of type B using an arbitrary (but finite) number of times an input of type A.

Thanks to its computational features, linear logic has established itself as an important tool for Implicit Computational Complexity (ICC), the branch of computational complexity studying languages or calculi able to capture the inherent principles of bounded computation without depending on specific machine models or explicit resource bounds. In particular, several variants of second-order linear logic called light logics have been proposed to capture complexity classes: examples are soft linear logic (SLL) [11] or light linear logic (LLL) [10] for FP (the class of polynomial time computable functions), and elementary linear logic (ELL) [5] for FELEMENTARY (the class of elementary time computable functions).

Continuing this tradition, in [14] Mazza and Terui introduced parsimonious logic, nuPLdf, a lambda calculus with polymorphic types inspired by linear logic that characterises the complexity class P/poly (the class of problems decidable in non-uniform polynomial time). In this system, the exponential modality satisfies the so-called Milner’s law !A ≃ A ⊗ !A. According to the Curry-Howard interpretation, this law allows us to interpret a formula !A as the type of streams over data of type A. Therefore, the linear implications A ⊗ !A  A (co-absorption) and !A  A ⊗ !A (absorption) can be respectively interpreted as the push and the pop operations on streams. In nuPLdf, non-uniformity is introduced by the typing rule !I, which takes a finite set of proofs D1, . . . , Dn of A and a (possibly non-recursive) function f : N −→ {1, . . . , n} as premises, and constructs a proof of !A modelling the stream Df(0) :: Df(1) :: Df(2) :: . . . . Specifically, the typing rule !I allows the encoding of advices for Turing machines, the crucial step to show completeness for P/poly. On the other hand, polynomial step cut-elimination is guaranteed thanks to “parsimony”, which invalids the implications !A  !A (digging) and !A  !A ⊗ !A (contraction).

1Formally, P/poly can be defined as the class of problems decidable by families of circuits with polynomial size or, equivalently, as the class of problems decidable in polynomial time by a Turing machine with polynomial advice, that is, an extra input whose size depends on the length of the input, but not on the input itself.
**Contribution**  In this talk we present an ongoing work exploring a different approach to \( \mathsf{P/poly} \) based on *Cyclic Implicit Complexity*, the study of ICC in the context of non-wellfounded proof theory \([3, 4]\). Specifically, the typing rule \( !I \) parametrised by a (possibly non-recursive) function \( f : \mathbb{N} \to \{1, \ldots, n\} \) and defining the stream \( D_f(0) :: D_f(1) :: D_f(2) :: \ldots \) will be represented by a non-wellfounded proof of the following form:

\[
\begin{array}{c}
\vdots \\
D_f(0) \\
\vdots \\
A \\
\vdots \\
!A \\
\vdots \\
c/p \\
A \\
\vdots \\
c/p \\
!A \\
\vdots \\
c/p \\
!A \\
\vdots \\
c/p \\
!A \\
\vdots \\
c/p \\
!A \\
\vdots \\
\end{array}
\]

essentially by “unpacking” \( !I \) into an infinite proof iterating a more primitive rule called *conditional promotion* (\( c/p \)).

The resulting system of non-wellfounded proofs for parsimonious logic, called \( \nu\mathsf{PLL}^\omega_2 \), introduces fallacious reasoning. Logical consistency is then recovered by adapting a standard global condition, called *progressiveness criterion*, which relies on threads of exponential formulas occurring in the infinite branches of a derivation tree. In particular, progressiveness forces a computational interpretation of the modalities \( ! \) and \( ? \) (i.e. the dual of \( ! \)), in terms of greatest and least fixed points respectively. Note that definitions of exponentials based on fixed points have been proposed in \([2]\) by defining \( !A := \nu\alpha. (\bot \& A \& (\alpha \& \alpha)) \) and \( ?A := \mu\alpha. (\bot \oplus A \oplus (\alpha \& \alpha)) \), where \( \nu \) and \( \mu \) are the greatest and the least fixed point operator respectively. However, as shown in \([5]\), such an encoding does not give rise to a Seely category, which is essential to model linear logic.

We discuss an alternative technique to prove cut-elimination for \( \nu\mathsf{PLL}^\omega_2 \) relying on infinitary rewriting techniques (see, e.g. \([7]\)), but avoiding the use of the multicut rule, as opposed to \([1]\). Then, we show that \( \nu\mathsf{PLL}^\omega_2 \) captures the class \( \mathsf{FP/poly} \) (the class of functions computable in non-uniform polynomial time). To this end, we establish a polynomial “modulus of continuity” for cut elimination (see e.g. \([13]\)), from which we infer soundness for \( \mathsf{FP/poly} \). This is one of the major technical results of the paper. Completeness is established via an encoding of polynomial time Turing machines with (polynomial) advice by adapting standard methods from \([12, 8]\) to the setting of non-uniform computation. Along the way, we show that \( \mathsf{cPLL}^\omega_2 \), i.e. the restriction of \( \nu\mathsf{PLL}^\omega_2 \) to proofs having a regular tree structure (known as circular proofs), captures precisely \( \mathsf{FP} \).

We conclude by introducing a relational semantics for \( \nu\mathsf{PLL}^\omega_2 \) and by analysing its interplay with cut elimination.

**References**


[12] Harry G. Mairson and Kazushige Terui. On the computational complexity of cut-
elimination in linear logic. In Carlo Blundo and Cosimo Laneve, editors, Theoreti-
cal Computer Science, 8th Italian Conference, ICTCS 2003, Bertinoro, Italy, Oc-
tober 13-15, 2003, Proceedings, volume 2841 of Lecture Notes in Computer Sci-

lambda-calculus. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias
Koutsoupias, editors, Automata, Languages, and Programming - 41st Interna-
tional Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Pro-
ceedings, Part II, volume 8573 of Lecture Notes in Computer Science, pages 305–

[14] Damiano Mazza. Simple parsimonious types and logarithmic space. In Stephan
Kreutzer, editor, 24th EACSL Annual Conference on Computer Science Logic,