An implicit characterisation of the polynomial hierarchy in an unbounded arithmetic

Patrick Baillot Anupam Das

Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP

1 Introduction and motivation

Today, there are countless approaches towards characterising complexity classes via logic. Foremost amongst these lies the proof-theoretic approach, characterising classes as the 'representable' functions of some logic or theory. Examples include bounded arithmetic [6] [13] [9], applicative theories [7] [12], intrinsic and ramified theories [16] [4], fragments of linear logic [11] [10] [14] [1] and fragments of intuitionistic logic [15].

To some extent there is a distinction between various notions of 'representability', namely between logics that *type* terms computing functions of a given complexity class, and theories that prove the *totality* or *convergence* of programs computing functions in a given complexity class. A somewhat orthogonal distinction is whether the constraints on the logic or theory are *implicit* or *explicit*. The former includes constraints such as ramification, type level and substructural considerations, while the latter includes bounded quantification, bounded modalities etc. This distinction is also naturally exhibited in associated function algebras, e.g. Cobham's *limited* recursion on notation [8] vs. Bellantoni and Cook's *predicative* recursion on notation [3].

While implicit constraints may be preferable since no bounds occur in the characterisation itself *per se*, explicit bounds are typically far more useful for more fine-grained characterisations of complexity classes. For instance, the polynomial hierarchy, **PH**, and its levels can be neatly characterised by the theories S_2^i of *bounded arithmetic*, using bounds on quantifiers to control complexity [6].[1]

In this work we improve the situation by using implicit methods in first-order theories to characterise **PH**. To achieve this we work with a function algebra of Bellantoni from [5] in which to extract programs, and use the *witness function method* of Buss to extract programs at ground type and preserve quantifier information, necessary to delineate the levels of **PH**.

2 Methodology

We describe some of the techniques used in this work, before briefly giving our main results in the next section.

[1] Other approaches to **PH** exist, but also use explicit bounds, e.g. [9], [12].

2.1 Implicit programs for PH

Extracting programs at ground type seems to be a necessity in order to delineate the levels of **PH**.[2] Therefore the natural programs in which to extract our witnesses will come from *recursion theoretic* characterisations. Of these, only the Bellantoni framework from [5], which extends safe recursive programs [3] by *predicative minimisation* constitutes an implicit characterisation, and so we extract our programs into this function algebra, henceforth denoted μBC.

2.2 Constraints on induction

An appealing feature of the bounded arithmetic approach is that bounds on (bounded) quantifier alternation in induction formulae precisely delimit the levels of **PH**, and we are able to replicate this property, only for unbounded quantifiers. Naturally, another constraint is required to stop ourselves from exhausting the arithmetical hierarchy once bounds are dismissed, and for this we use essentially a *ramification* of individuals: explicit predicates $N_0, N_1, \ldots$ are used similarly to Peano's N predicate to intuitively indicate 'how sure' we are that a variable denotes a genuine natural number.

In fact, two predicates suffice and their relationship is entirely governed by the equation $N_1(x) \iff \Box N_0(x)$, under the laws of the modal logic *S4*. The distinction between the two predicates corresponds to the distinction between safe and normal variables in BC programs, which was an observation from previous work [2]. A similar phenomenon occurs in Cantini's work [7], which presents a characterisation of **P** in an *applicative theory*, in order to extract BC programs. While he allows arbitrary alternation of unbounded quantifiers, note that his induction is *positive*, and so universal quantifiers cannot vary over certified natural numbers, i.e. individuals in N. In fact this sort of unbounded quantification is also compatible with our approach of [2].

The idea of using ramified theories in implicit complexity, inducting on normal variables rather than safe, is largely due to Leivant [16], who introduced *intrinsic theories* for an arbitrary free algebra. The same idea was also used in [17] for a theory of arithmetic characterising the elementary functions, somewhat reworking of Leivant's work. The main difference in this work, which is why we are able to characterise **PH**, is that for our model of computation we simply use formulas of the theory rather than equational specifications used in [16] and [17]. This indeed has a significant effect on the complexity class characterised, as observed in [4].

2.3 Extraction at ground type

We rely on the *witness function method* for extracting functions at bounded type. The idea is as follows:

1. Reduce a proof to *De Morgan* normal form, with formulae over the basis $\{\bot, \top, \vee, \wedge, \exists, \wedge\}$ and negation restricted to atoms.

2. Conduct a *free-cut elimination* on the proof, resulting in a proof whose formulae are restricted to essentially just subformulae of the conclusion, axioms and nonlogical steps.

[2] Indeed we are not aware of any 'higher-type' characterisation of **PH**.

3. Extract witnesses inductively from the proof into an appropriate function algebra, verifying the necessary semantic properties along the way.

1 ensures that our extraction works at ground type, rather than higher types which are typically necessary when negation has larger scope. At the same time it preserves the quantifier alternation information that is crucial to distinguishing the levels of **PH**. 2 allows us to assume that all formulae in a proof have logical complexity bounded by that of induction formulae. This means that, when extracting programs via 3, quantifier alternation of induction formulae corresponds to the depth of minimisation operators in a μBC program, allowing for a level-by-level correspondence with the polynomial hierarchy.

2.4 Completeness for PH

In the other direction, showing completeness for **PH**, we are able to formalise a more-or-less standard argument, e.g. from bounded arithmetic [6], where applications of minimisation in a program correspond to applications of the *well ordering property* in arithmetic. This is in turn is a corollary of induction but, in this case, crucially relies on the use of *right-contraction* in the logic. It seems that this feature is crucial in distinguishing these theories from 'linear' variants like in previous work [2], and in particular work of Bellantoni and Hofmann [4] where, without right-contraction, any number of quantifier alternations still corresponds to only polynomial time computation.

3 Results

Due to space constraints, we give only an informal account of our results.

We define a family of theories B^i over the language $\{0, s, +, \times, \#, |\cdot|, \lfloor \frac{\cdot}{2} \rfloor\}$ of bounded arithmetic augmented with two predicates, N_0 and N_1, which act as ramifiers for our theory. These replicate the distinction between safe and normal input, respectively, for μBC programs. We set $N_1 \subseteq N_0$ and crucially have the following non-logical inference step:[3]

$$\frac{\vdash \forall \vec{x} \in N_1 . \exists \vec{y} \in N_0 . A(\vec{x}, \vec{y})}{\vdash \forall \vec{x} \in N_1 . \exists \vec{y} \in N_1 . A(\vec{x}, \vec{y})}$$

For the arithmetic function symbols we give their basic defining axioms as for the theories S^i_2 from [6], only relativising quantifiers to match their safe-normal sorting of arguments from the Bellantoni-Cook framework.[4]

We define the $\Sigma_i^{N_0}$-$\Pi_i^{N_0}$ formula hierarchy analogously to the (bounded) arithmetical hierarchy, only counting alternations of 'safe quantifiers', i.e. those relativised to N_0. The theory B^i_2 contains the axiom schema of *normal polynomial induction* on $\Sigma_i^{N_0}$ formulae A:

$$\left(A(0) \wedge \forall u \in N_1 . (A\left(\lfloor \tfrac{u}{2} \rfloor\right) \supset A(u))\right) \supset \forall u \in N_1 . A(u)$$

Our main result is the following:

Theorem. *B^i_2 proves the totality of precisely the $\Box^p_i$ functions.*

[3] Seen from the modal point of view, this is a form of *necessitation* for models whose second-order parts are closed under first-order comprehension.

[4] For instance, $s, +, |\cdot|, \lfloor \frac{\cdot}{2} \rfloor$ take N_0 arguments, $\times$ takes one N_0 and one N_1, and $\#$ takes two N_1 arguments. All terms are in N_0.

References

[1] Patrick Baillot. On the expressivity of elementary linear logic: Characterizing ptime and an exponential time hierarchy. *Inf. Comput.*, 241:3–31, 2015.

[2] Patrick Baillot and Anupam Das. Free-cut elimination in linear logic and an application to a feasible arithmetic. In *Proceedings of CSL 2016*, volume 62 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

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

[4] Stephen Bellantoni and Martin Hofmann. A new "feasible" arithmetic. *J. Symb. Log.*, 67(1):104–116, 2002.

[5] Stephen J. Bellantoni. *Predicative Recursion and Computational Complexity.* PhD thesis, University of Toronto, 1992.

[6] Samuel R Buss. *Bounded arithmetic*, volume 86. Bibliopolis, 1986.

[7] Andrea Cantini. Polytime, combinatory logic and positive safe induction. *Arch. Math. Log.*, 41(2):169–189, 2002.

[8] A. Cobham. On the intrinsic computational difficulty of functions. In *Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, pages 24–30. North Holland, Amsterdam, 1964.

[9] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[10] Jean-Yves Girard. Light linear logic. In *Logical and Computational Complexity. Selected Papers. LCC '94.*, pages 145–176, 1994.

[11] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.

[12] Reinhard Kahle and Isabel Oitavem. Applicative theories for the polynomial hierarchy of time and its levels. *Ann. Pure Appl. Logic*, 164(6):663–675, 2013.

[13] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory.* Cambridge University Press, New York, NY, USA, 1995.

[14] Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.

[15] Daniel Leivant. A foundational delineation of poly-time. *Inf. Comput.*, 110(2):391–420, 1994.

[16] Daniel Leivant. Intrinsic theories and computational complexity. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 1995.

[17] Geoffrey E. Ostrin and Stanley S. Wainer. Elementary arithmetic. *Ann. Pure Appl. Logic*, 133(1-3):275–292, 2005.