Towards an implicit characterisation of the polynomial hierarchy in an unbounded arithmetic

Work in progress

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 [7] [17] [12], applicative theories [9] [16], intrinsic and ramified theories [20] [5], fragments of linear logic [15] [14] [18] [2] and fragments of intuitionistic logic [19].

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. Perhaps a more important (and somewhat orthogonal) distinction for the Dice-Fopara community 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 [11] vs. Bellantoni and Cook's *predicative* recursion on notation [4].

Some correlations abound: explicit bounds are typically far more useful for more fine-grained characterisations of complexity classes, e.g. levels of the polynomial or arithmetical hierarchies, and often admit witness extraction methods that remain in a ground type programming language, e.g. via recursion theoretic characterisations. Implicit bounds, however, are more often associated with higher-typed programming languages, which are arguably more useful for achieving witness extraction at all for powerful theories such as arithmetic and set theory, cf. [1], [22]. Complexity bounds are harder to obtain, but the framework is nonetheless somewhat more desirable since no bounds occur in the characterisation itself *per se*.

In this line of work we attempt to ameliorate the situation by using implicit methods to delineate fine-grained hierarchies of feasible complexity classes, namely the *polynomial hierarchy*, **PH**. One particular feature of this work that helps make this possible is to break one of the aforementioned correlations: while we use implicit constraints, our witness extraction methods will use only

functions of bounded type level. In this way we can naturally appeal to function algebras, which are of ground type in nature, which implicitly characterise **PH**, namely via *predicative minimisation* [6]. In order to remain in this class of programs and not get lost with higher types, we appeal to the *witness function method* of extracting programs from proofs, a technique developed by Buss [7] [8], which is ideal for extracting ground programs directly from classical proofs in weak theories. This extends work presented in [3].

2 State of the art

As we have already argued, it is natural to expect that characterisations of hierarchies such as **PH** are more readily established by using ground or bounded type witness extraction procedures, due to the correspondence between logical searches in a program and the quantification over objects of ground type in a theory. As justification for this position, consider the following table of examples of known characterisations:[1]

Class	Ground	Higher order
$\mathbf{NC}^i$	TNC^i [10], VNC^i [12]	-
P	S_2^1 [7], [21], V^1 [23] [12]	LLL [14], SLL [18]
$\Box_i$	S_2^i [7], [16]	-
PH	S_2	-
PSPACE	U_2^1 [7]	STA_B [13]
Elementary	$I\Delta_0 + \exp$	ELL [14]

Thus, if we want an implicit characterisation of **PH** in a logical theory, we should break the apparent (although not universal) link between 'implicit' and 'higher type'. Now, if we zoom in on the ground setting, where extracted programs do not make use of higher types, there are still several parameters by which the characterisations can vary, in particular:

- How are programs specified in the language of the theory? By a formula, as in Peano arithmetic, by a first-order equational program, or by an applicative term in the style of combinatory algebra;
- What type of programs are extracted from proofs of the theory? A program of a bounded recursion class, e.g. of Cobham's algebra, or of a tiered recursion class, e.g. of Bellantoni and Cook's algebra.

We classify some known characterisations from the literature according to these two parameters in the following table:

	bounded rec. programs	tiered rec. programs
formula	**PH**, $\Box_i$ (Buss [7])	
equational		**P** (Leivant [20])
applicative	**P** (Strahm [21]) **PH** (Kahle-Oitavem [16])	**P** (Cantini [9])

[1] All classes can be taken in their functional variations.

Our goal is to extend the approach using extraction of programs of a tiered recursion class (second column), using the formula style of specification (first row), to the whole of the polynomial hierarchy, i.e. **PH** and its levels $\Box_i$.

3 Towards an implicit theory for PH

We explain our approach for this work-in-progress in more detail here, referring to previous work when analogous methods are used.

3.1 Implicit programs for PH

Since we want to remain at ground type, the natural programs in which to extract our witnesses will come from recursion theoretic characterisations, cf. the table above. Indeed, as we have already mentioned, we are not aware of any 'higher-type' characterisation of **PH**. Of these, only the Bellantoni framework from [6], which extends BC-programs by *predicative minimisation* constitutes an implicit characterisation, and so we will look to extract our programs into this function algebra, henceforth denoted μBC.

3.2 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 one of our desiderata is to replicate this property, only for unbounded quantifiers. Naturally, another constraint will be 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 $N_0, N_1, \ldots$ will be 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 will 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-like programs, which was an observation from previous work [3]. A similar phenomenon occurs in Cantini's work [9], 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 [3].

3.3 Extraction at ground type

As we did in [3], we will 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.

3. Extract witnesses inductively from the proof, with appropriate semantic properties of these programs verified by an interpretation into a (classical) quantifier-free theory.

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, and so potentially allows for a level-by-level correspondence with the polynomial hierarchy.

We point out that, in some ways, this is similar to approaches from applicative theories, which typically use free-cut elimination followed by a direct *realisability* argument, e.g. in [21], [9] and [16]. Indeed this could have been possible in our previous work [3], as Cantini did in his work [9], for a characterisation of **P**. However, in this case, since the quantifiers are unbounded the realisability argument is apparently not readily formalised, and it is therefore quite natural to pursue a *bona fide* proof interpretation.

3.4 Completeness for PH

In the other direction, showing completeness for **PH**, it seems straightforward to formalise a standard argument, e.g. from bounded arithmetic [7], where applications of minimisation in a program correspond to 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 [3], and in particular work of Bellantoni and Hofmann [5] where, without right-contraction, any number of quantifier alternations still corresponds to only polynomial time computation.

3.5 Putting it all together

To summarise the main goal of this work-in-progress, we are aiming for a variation of the following result:

Conjecture. *First-order classical modal logic S4 with induction restricted to non-modal formulae, over a suitable set of axioms, characterises the class* **PH**. *Bounds on quantifier alternation in induction formulae delimit the levels of* **PH**.

4 Conclusions

We surveyed the state of the art for representing function classes proof theoretically by logics and theories, and considered the problem of finding an implicit characterisation of **PH**. Identifying the witness function method as a useful tool for witness extraction at bounded type level, a seemingly important prerequisite for characterising **PH**, we sought to calibrate an appropriate theory of arithmetic for witness extraction to the μBC characterisation of **PH**. We

presented a conjecture that a modal theory suffices to carry out this characterisation, based on previous work by ourselves and others[3] [9] [5] and proving this result constitutes the outstanding work-in-progress.

References

[1] Jeremy Avigad. Gödels functional (Dialectica) interpretation. *Handbook of Proof Theory*, 137, 1998.

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

[3] 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.

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

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

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

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

[8] Samuel R Buss. The witness function method and provably recursive functions of peano arithmetic. *Studies in Logic and the Foundations of Mathematics*, 134:29–68, 1995.

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

[10] Peter Clote and Gaisi Takeuti. First order bounded arithmetic and small boolean circuit complexity classes. In *Feasible Mathematics II*, pages 154–218. Springer, 1995.

[11] 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.

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

[13] Marco Gaboardi, Jean-Yves Marion, and Simona Ronchi Della Rocca. An implicit characterization of PSPACE. *ACM Trans. Comput. Log.*, 13(2):18:1–18:36, 2012.

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

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] Thomas Strahm. Theories with self-application and computational complexity. *Inf. Comput.*, 185(2):263–297, 2003.

[22] Anne Sjerp Troelstra. Realizability. *Handbook of Proof Theory*, 1998.

[23] Domenico Zambella. Notes on polynomially bounded arithmetic. *J. Symb. Log.*, 61(3):942–966, 1996.