Bounded Arithmetic with Bounded Finite Types

K. Ojakian

Instituto Superior Técnico, Lisbon (Portugal)

This abstract describes work in progress. Bounded arithmetic has typically been done in the first order context with numbers as the intended objects and in the second order context, which includes finite sets of natural numbers as intended objects. Parikh [4] originally developed bounded arithmetic as a first-order theory. Buss [1] extended this work and also introduced second-order bounded arithmetic. Zambella [6] and Cook [2] have introduced modified versions of these second-order systems. Skelley [5] introduced a third-order theory, which includes as intended objects the finite sets of finite sets of natural numbers. We extend this further by allowing "bounded objects of finite type," meaning that we consider all the finite types, but restrict the objects to just the bounded (or finite) objects of this type. The motivation for this development is twofold. For each of the theories (first, second, and third order), a number of similar theorems are proved in analogous ways (for example the correspondence to polynomial time functions). In our context it seems that we can give one proof and then note the results for various systems as relatively simple corollaries. The second goal is to provide a system in which proofs can be carried out in a more transparent manner. Tied in with this goal is the motivation of providing a good context for formalizing finite combinatorics and developing its "Reverse Mathematics." Bounded Arithmetic is a natural context for finite combinatorics due to its limited strength, providing us with theories that correspond to relatively weak complexity classes, such as polynomial time. We now give a stream-lined discussion of some of the technical development, just to give an idea of how it works. The development is motivated by the work of Ferreira [3].

We define the types inductively:

ω is a type, and if α and β are types, then $(\alpha \to \beta)$ is a type.

The intended objects of type ω are the natural numbers. The intended objects of type $(\alpha \to \beta)$ are, roughly, the finite functions from the finite objects of type α to the finite objects of type β. For each type, we in fact define a linear order on those objects. A function f of type $(\alpha \to \beta)$ has an associated object d of type α, such that f is defined for $x < d$ of type α; i.e. d is used to specify the domain of f.

There is the well-known notion of degree which categorizes types roughly in terms of how big they are (assuming you start with the same size objects of type ω). We define the degree of a type (a natural number), inductively:

$$\text{degree}(\omega) = 0 \text{ and } \text{degree}(\alpha \to \beta) = \max(\text{degree}(\beta), 1 + \text{degree}(\alpha)).$$

We now say some words about the terms. For each type τ, we have the empty function of that type, denoted by the term $\mathbf{0}^\tau$. We will have term building operations which allow objects to grow at exactly the right growth rate: That is, an object can grow in size exactly as much as it could grow if it were the input to a polynomial time Turing Machine. In particular we use a binary operation similar to Ferreira's string operation $\times$, which grows like Buss' smash function. We also have a concatenation operation. Given a finite function f of type $\alpha \to \beta$ and an object b of type β, $f \frown b$ is the function f with its domain extended by one object of type α, where this new object is mapped to b. Given a term f of type $\alpha \to \beta$, **dom** (f) is a term of type α, which gives the domain in the sense that f is defined for $x < \mathbf{dom}(f)$.

Formulas are defined in a standard way, starting with atomic formulas of the form $(t = s)$ or $(t < s)$, where s and t are terms of the same type; we build formulas in the usual way using $\wedge$, $\vee$, $\neg$, $\forall$, and $\exists$. Bounded quantifiers are defined in the usual way for any type, using the linear order. For example, $\exists x^\tau \leq t^\tau\ \psi$ abbreviates $\exists x^\tau\ ((x \leq t^\tau) \wedge \psi)$, and supposing τ is of degree k, it is called a degree k bounded quantifier. If the bound t in the quantifier is the term $\mathbf{dom}(s)$, where s is a term of degree k, then the quantifier is called a sharply bounded quantifier of degree k. As is typical in bounded arithmetic, we can now define classes of formulas by counting quantifier alternations of the appropriate bounded quantifiers. Let $\Sigma_i^{b,(k)}$ be the class of formulas with $\leq i$ degree $\leq k$ bounded quantifier alternations (beginning with an $\exists$), any number of sharply bounded quantifiers of degree $\leq k$, and no quantifiers of larger degree.

To define our theories we will have a number of basic axioms about the basic operations (a finite number *for each type*); we call these axioms BASIC. We will also have induction axioms. Suppose $\psi(x^{\alpha\to\beta})$ is a formula. By $\psi-\mathsf{CAT}$ we mean the following "concatenation" induction axiom:

$$\left(\begin{array}{c} \psi(\mathbf{0}^{\alpha\to\beta}) \\ \bigwedge \\ \forall f^{\alpha\to\beta}\ \forall r^{\beta} \quad \psi(f) \Rightarrow \psi(f \frown r) \end{array} \right) \quad \Rightarrow \quad \forall f^{\alpha\to\beta}\ \psi(f)$$

We then define $\mathsf{S}^{i,(k)}$ to be the theory defined by the following set of axioms:

BASIC, together with $\psi-\mathsf{CAT}$, for all $\Sigma_i^{b,(k)}$ formula ψ.

To make the connection to polynomial time, notice that for any type, we can think of some standard way of coding its objects as a sequence of symbols for input to a Turing Machine. However, we also develop a notion of polynomial time computation on these objects using random access machines for finite types. Though either approach works, the latter one provides a natural way to think about the higher type objects. The main claim is then to connect the polynomial functions on the higher type objects to these theories:

The polynomial functions on degree k objects are exactly the $\Sigma_1^{b,(k)}-$definable functions in $\mathsf{S}^{1,(k)}$.

References

[1] S. Buss, *Bounded Arithmetic*, Bibliopolis (1986).

[2] S. Cook, *CSC 2429S Notes*, course notes on web site (2002).

[3] F. Ferreira, *A feasible theory for analysis.* The Journal of Symbolic Logic, 59 (1994), pp. 1001–1011.

[4] R. Parikh, *Existence and feasibility in arithmetic.* The Journal of Symbolic Logic, 36 (1971), pp. 494–508.

[5] A. Skelley, *A Third-Order Bounded Arithmetic Theory for PSPACE*, In Computer Science Logic: 18th International Workshop of the EACSL, Springer LNCS (2004), pp. 340–354.

[6] D. Zambella, *Notes On Polynomially Bounded Arithmetic.* The Journal of Symbolic Logic, 61 (1996), pp. 942–966.