Third-Order Computation and Bounded Arithmetic

Alan Skelley[*]

Mathematical Institute, Academy of Sciences of the Czech Republic
Žitná 25, CZ - 115 67 Praha 1, Czech Republic
skelley@math.cas.cz

Abstract. We describe a natural generalization of ordinary computation to a third-order setting and give a function calculus with nice properties and recursion-theoretic characterizations of several large complexity classes. We then present a number of third-order theories of bounded arithmetic whose definable functions are the classes of the EXP-time hierarchy in the third-order setting.
Keywords: bounded arithmetic, recursion theory, computability, computational complexity

1 Introduction

Bounded arithmetic is an important and useful way to approach problems in computational and propositional proof complexity: strong tools from logic and model theory can be applied, and many of the connections are intriguingly not tight, suggesting that it could be possible to skirt around the major barriers of complexity theory. The second-order viewpoint of Zambella and Cook associates second-order theories of bounded arithmetic with various complexity classes by studying the definable functions of strings, rather than numbers. This approach simplifies presentation of the theories and their propositional translations, and furthermore is applicable to complexity classes that previously had no corresponding theories.

In previous work [12], we adapted the second-order viewpoint to PSPACE with the third-order theory W_1^1. In what follows, we generalize this result in several directions: First, by expanding the notion of computation to the third-order setting, essentially allowing a natural way to compute with very large objects, admitting a function calculus with nice properties and obtaining useful recursion-theoretic characterizations of large complexity classes above PSPACE. This computational setting bridges a gap by simultaneously allowing more natural reasoning about the kind of computation captured by theories of bounded arithmetic, while at the same time remaining a natural extension of ordinary computation and complexity. The second direction of generalization is to a full hierarchy of theories for the EXP-time hierarchy in this general setting. We also

[*] Partially supported by a Canadian NSERC postdoctoral fellowship

show how to apply the recursion-theoretic characterization of PSPACE to obtain a "minimal" theory for that class.

The remainder is organized as follows: In section 2 we describe the third-order setting: first the framework for bounded arithmetic, then for computation, and discuss complexity and recursion theory. Section 3 presents theories of bounded arithmetic and results about definability. We conclude with some open problems.

2 The Third-Order Setting

2.1 Bounded Arithmetic

The three sorts of third-order bounded arithmetic are intended to represent natural numbers, finite sets of natural numbers, and finite sets of such sets. For free and bound variables of these sorts we respectively use $a, b, c, ...$ and $x, y, z, ...$; $A, B, C, ...$ and $X, Y, Z, ...$; and $\mathcal{A}, \mathcal{B}, \mathcal{C}, ...$ and $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, ...$ The language $\mathcal{L}_A^3 := \{0, 1, +, \cdot, |\cdot|_2, \in_2, \in_3, \leq, =_1, =_2\}$ of nonlogical symbols is the same as the set $\mathcal{L}_A^2$ for V^1 but with the addition of the third-order membership predicate $A \in_3 \mathcal{B}$; note the absence of smash ('#') and third-order equality or length. We write $\mathcal{A}(B)$ for $B \in_3 \mathcal{A}$ and similarly for $\in_2$. The second sort closely represent finite binary strings as in e.g. [5] and likewise with the third sort (with strings rather than numbers as bit-indices), so we refer to them respectively as "strings" and "superstrings". We use a tilde: $\tilde{x}$ to denote unspecified sort.

There is a hierarchy of classes $g\Sigma_i^{\mathcal{B}}$ and $g\Pi_i^{\mathcal{B}}$ of formulas in this language analogous to the hierarchies Σ_i^B and Π_i^B of second-order formulas: the subscript counts alternations of third-order quantifiers (bounded string and number quantifiers ignored) and the 'g' denotes general (not strict) quantifier syntax. Note that there is no way to bound third-order quantifiers, but the number and string parameters determine the number of initial bits of superstrings that are relevant to the truth-value of a formula.

2.2 Computation

Our intent now is to capture the nature of string-based computation defined by third-order theories of bounded arithmetic. For this reason, our primary focus is on classes of polynomially-bounded functions (from strings to strings) or similar, as this makes operations such as composition of functions more natural and matches ordinary complexity theory. We are consequently interested in our classes of functions somehow maintaining an exponential-size distinction between the three sorts, as do (standard) theories of bounded arithmetic. Furthermore, our intent when defining third-order complexity classes is that the third-order (superstring) arguments not count towards the resource limits of the machine.

Functions in our setting will be strongly typed (each function has a fixed signature specifying sorts of arguments and output). The domains of the three sorts are: $\mathbb{D}_1 := \mathbb{N}$; $\mathbb{D}_2 := \{S \subset \mathbb{N} : |S| < \infty\}$; and $\mathbb{D}_3 = \{\mathcal{S} \subset \mathbb{D}_2 : |\mathcal{S}| < \infty\}$. Again we shall refer to these as numbers, strings, and superstrings; these sorts are the same as the intended interpretations of the three sorts of variables in third-order bounded arithmetic and we use similar notation. Function symbols in our calculi will similarly be named $f, g, ...; F, G, ...;$ and $\mathcal{F}, \mathcal{G}, ...$ to indicate the sort of **the range of the function**. Let $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2 \cup \mathbb{E}_3$ be the set of all functions of fixed signature, categorized according to the sort of the output. The 0-1 valued functions (predicates) are referred to as $\mathbb{E}_0 \subset \mathbb{E}_1$.

Such functions are computed by Turing machines or other computational models by receiving number inputs in unary, strings as usual, and superstrings by random access. Outputs of strings or numbers are the same way, while superstrings are either output on a write-only tape, or "by query", as a predicate with a distinguished string input as the characteristic function of the output bits of the superstring-valued function. Precise definitions are in [13]. We are interested primarily in **polynomially bounded** functions. In the context of third-order computation, we mean that the polynomial bound applies to the value of a number output or the length of a string output, and is computed using only the number inputs and the lengths of the string inputs. If there are only superstring inputs, then the bound is a constant, and every superstring-valued function is polynomially bounded.

2.3 Complexity

An ordinary function or language class becomes a complexity class of third-order functions as follows: The notation (various superscripts on the complexity classes) is: For FC a function class, FC^+ is the third-order class with superstring output on write-only tape; for C a class of languages, $C^\diamond$ is the class of third-order predicates, while C° are the functions computed "by query" by predicates in this class. Here we describe some specific cases of complexity classes we are interested in:

First, FPSPACE$^+$ is the third-order analogue of PSPACE functions. It consists of those polynomially bounded functions computable by a machine in polynomial space (as a function of the string and (unary) number inputs only), where superstring outputs are written onto a write-only output tape, allowing exponential-length superstring outputs. The machine's queries to its superstring inputs must also be polynomially bounded (as a function of its inputs). FEXP$^+$ is similarly the polynomially bounded exponential-time functions with polynomially bounded access to superstring inputs. In contrast to FPSPACE$^+$, the polynomial bound is an actual restriction as an exponential time machine could otherwise write exponentially large strings (either as output, or as a query to superstring inputs).

Now for the case of polynomial time, the class FP$^+$ defined analogously to FPSPACE$^+$ and FEXP$^+$ has the property that superstring outputs have polynomial length, due to the time bound of the machines; however, the class P$^\circ$ of polynomially-bounded functions computed by "by query" by polynomial-time machines does not have this restriction. For this reason, FP$^+ \cup$P$^\circ$ is in some contexts a more suitable third-order analogue of P. This is also the case for functions from levels $\square_i^p$ of the polynomial-time hierarchy, which are computed by polynomial-time machines with access to an oracle from Σ^p_{i-1}: The third-order class $(\Box^p_i)^+$ is restricted to polynomially many bits in its superstring outputs and so $(\Box^p_i)^+ \cup (\Box^p_i)^\circ$ is a more appropriate definition.

As a final set of examples, the predicate classes P$^\diamond$, NP$^\diamond$, $(\Sigma^p_i)^\diamond$, NEXP$^\diamond$ and $(\Sigma^{exp}_i)^\diamond$ are 0-1 valued functions, and are the characteristic functions of machines from the corresponding ordinary complexity classes, modified with polynomially bounded access to superstring inputs.

Some comments are in order concerning these classes. First, and most importantly, the third-order complexity classes discussed thus far, restricted to functions from strings to strings (or string predicates) are the usual complexity classes. There are nevertheless some interesting observations to be made: For example P$^\diamond$ $\neq$NP$^\diamond$, as a predicate in the latter class can determine if a given superstring contains a 1 (up to a bound given by a string argument), while this predicate is clearly not in P$^\diamond$. The usual argument for Savitch's theorem goes through, at least for (unrelativized) NPSPACE$^\diamond$: configurations are still described by polynomial-sized strings, including queries to superstring inputs. We conclude that PSPACE$^\diamond$ =NPSPACE$^\diamond$.

Now, in order to expand our discussion to the exponential-time hierarchy, we must first address relativizing classes of functions by adding oracles in the form of access to a third-order function. Formally, a **third-order oracle** Turing machine has a number of specified write-only query tapes, each one designated with a sort. The machine may write values on these tapes which are polynomially bounded, in the sense that the numbers (in unary), and lengths of strings written are all bounded by fixed polynomials in the machine's (non-superstring) inputs. When the machine enters the special query state, these tapes are erased, and a value is returned to the machine by way of a special read-only reply tape (with random access in the case of a superstring-valued oracle).

The usual exponential-time hierarchy has definition $\Sigma^{exp}_i = \text{NEXP}^{\Sigma^p_{i-1}}$ [10]. This is equal to Σ_i-TIME(exp), which are the languages computed by exponential time alternating Turing machines with i alternations (starting with existential). Paralleling this definition, we can define the corresponding classes of 0-1 valued functions from $\mathbb{E}_0$. It is important to observe that the queries made of the Σ^p_{i-1} oracle by the NEXP machine in the standard definition are in general of **exponential** size. Our third-order oracle machines can also issue exponentially-long queries to their oracles, but these must be in the form of superstrings, as the string inputs to oracles are restricted to be polynomially bounded per our definition. Consequently the complexity class of the third-order oracle we use will be different.

We therefore define $(\Sigma^{exp}_1)^\diamond$ =NEXP$^\diamond$ and $(\Sigma^{exp}_i)^\diamond = (\text{NEXP}^\diamond)^{(\Sigma^{exp}_{i-1})^\diamond}$. In other words, each higher level of the hierarchy is obtained by augmenting nondeterministic exponential time with a third-order oracle for the previous level. Since the queries to this oracle must be polynomially bounded (although this still allows exponential-length superstring inputs to the oracle), it can be seen that this relativization corresponds to unbounded access to an ordinary oracle from the appropriate level of the **quasi-polynomial-time** hierarchy (considered as a predicate on the superstring inputs): For example, if an NEXP machine writes string and superstring inputs of lengths $p(n)$ and $2^{p(n)}$ respectively to a third-order NEXP oracle, then the query can be answered in nondeterministic time $2^{(p(n))^k}$ for some k, which is exponential in $p(n)$. In terms of the length of the superstring input, $2^{p(n)}$, the quantity $2^{(p(n))^k}$ is quasi-polynomial.

In the hands of an NEXP machine, however, an unbounded (ordinary) oracle from some level of the quasi-polynomial-time hierarchy is no more powerful than one from the same level of the polynomial-time hierarchy, as the machine could simply make longer queries (i.e. $2^{(p(n))^k}$) of this latter oracle. Thus as predicates purely on strings, the levels of our hierarchy correspond precisely with the levels of the ordinary exponential-time hierarchy. Therefore:

Theorem 1. *The predicates represented in the standard model by $\Sigma_0^{\mathcal{B}}$-formulas are precisely $PH^{\diamond}$; for $i \geq 1$ those represented by $g\Sigma_i^{\mathcal{B}}$- and $g\Pi_i^{\mathcal{B}}$-formulas (and also the strict versions of these classes) are precisely $(\Sigma_i^{exp})^{\diamond}$ and $(\Pi_i^{exp})^{\diamond}$, respectively.*

The function classes $(\square_i^{exp})^{+}$:=$(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^{\diamond}})^{+}$ are the polynomially bounded functions computed by exponential-time Turing machines relativized with a third-order oracle for a predicate from $(\Sigma_i^{exp})^{\diamond}$, and similarly as functions purely of strings correspond to the usual $\square_i^{exp}$. It should be noted that $(\Sigma_i^{exp})^{\diamond} = (\Pi_i^{exp})^{\diamond}$ seems to imply that the third-order exponential-time hierarchy collapses to the ith level, while this is not known for the ordinary case; The difference is that the assumption $\Sigma_i = \Pi_i$ in the third-order context is stronger, in that it covers also predicates on superstrings.

2.4 Recursion Theory of Functions

First some standard functions: The number functions $\{x + y, x \cdot y\}$, constants 0,1, etc. are as usual. The bit, string successor and concatenation functions $\{\mathrm{bit}(x, Y), s_0(X), s_1(X), X \frown Y\}$ are also standard, but they are operations on binary strings, while our string-like domain $\mathbb{D}_2$ consists of finite sets of natural numbers. We therefore define these functions to operate on the strings represented by the input finite sets, and to output the set representing the desired string. $\{|X|, X \in \mathcal{Y}, 1^x\}$ respectively give the least upper bound of the set (which is one more than the length of the string being represented by the set), the (0-1-valued) characteristic function of $\mathcal{Y}$, and a standard string of x bits (represented by a set of least upper bound $x + 1$). All of the functions described thus far are polynomially bounded.

We now define several operations on these functions. As our focus is on string functions as opposed to the standard recursion-theoretic viewpoint of number functions, we shall comment in each case on how these operations compare to standard operations on number functions.

First, the operation of **composition** defines a function $\tilde{f}(\tilde{\overline{x}}) = t$ by specifying a term t consisting of variables among $\tilde{\overline{x}}$ and other functions, constructed in such a way that arities and argument types are respected. Observe that this operation allows permutation and renaming of variables.

Define $\tilde{f}$ (of any sort) by **limited recursion** from $\tilde{g}$, $\tilde{h}$ (also of any sort) and l by $\tilde{f}(0,...) = \tilde{g}(...), \tilde{f}(x+1,...) = \tilde{h}(x, \tilde{f}(x,...),...)$ and either $\tilde{f}(x,...) \leq l(x,...)$ or $|\tilde{f}(x,...)| \leq l(x,...)$, as appropriate. This operation corresponds roughly to limited recursion on notation for number functions, as it iterates a function ($\tilde{h}$) a polynomial number of times subject to a bound on growth. **Recursion** is the same operation without the bound on growth.

Define $\tilde{f}$ by **limited doubling recursion** from $\tilde{g}$ and l by $\tilde{f}(0,\tilde{y},...) = \tilde{g}(\tilde{y},...), \tilde{f}(x+1,\tilde{y},...) = \tilde{f}(x, \tilde{f}(x,\tilde{y},...),...)$ and either $\tilde{f}(x,\tilde{y},...) \leq l(x,...)$ or $|\tilde{f}(x,\tilde{y},...)| \leq l(x,...)$, as appropriate. This operation corresponds roughly to limited recursion for number functions, as it iterates a function ($\tilde{g}$) an exponential number of times (by doubling the number of nestings a polynomial number of times) subject to a bound on growth. **Doubling recursion** is the same operation without the bound on growth.

Define $\tilde{f}$ (of any sort) by **limited long recursion** from $\tilde{g}$, $\tilde{h}$ (also of any sort) and l by $\tilde{f}(1^0,...) = \tilde{g}(...), \tilde{f}(X+1,...) = \tilde{h}(x, \tilde{f}(X,...),...)$ and either $\tilde{f}(X,...) \leq l(X,...)$ or $|\tilde{f}(X,...)| \leq l(X,...)$, as appropriate. This operation is similar to the previous one in that it iterates a function an exponential number of times; however, it differs in that the exponentially many iterations are performed directly by using a string as an exponential-length counter. This operation presupposes a suitable string successor function $X+1$.

Define $\mathcal{F}$ by **limited 3-comprehension** from $g, h \in \mathbb{E}_1$ by $\mathcal{F}(..)(X) \leftrightarrow (|X| \leq g(..) \wedge h(X,..) = 0)$.

It should be noted here that the recursion operations, as well as simple composition of functions, appear to be significantly more powerful when applied to superstring-valued functions. This is because in the composition of two such functions, the space may not be available to write down the intermediate value. A space-bounded computation model would then have to query the "inner" function many times (to retrieve bits of its output as needed) in order to compute the outer function. The composition of two polynomially bounded number- or string-valued functions can be computed using the sum of the time requirements (computing first one then the other function), while the required space does not increase. For superstring-valued functions, on the other hand, the time required for the composition as described seems in general to be the product of the time required for each component, while the space required is the sum. If space is not bounded then the intermediate results can be written in full, and thus time and space requirements are as for the composition of number- or string-valued functions.

At this point we can characterize several important complexity classes:

Theorem 2. *1. $FP^{+} \cup P^{\circ}$ is the closure of the initial functions $I = \{0, 1, x+y, x \cdot y, 1^x, |X|, s_0(X), s_1(X), bit(x,Y), X \frown Y, X \in \mathcal{Y}\}$ under composition, limited 3-comprehension and limited recursion with the latter restricted to $\mathbb{E}_1 \cup \mathbb{E}_2$.*

2. *FPSPACE$^+$ is the closure of I under composition, limited 3-comprehension and limited recursion.*
3. *FPSPACE$^+$ is the closure of I under composition, limited 3-comprehension and limited doubling recursion restricted to $\mathbb{E}_1 \cup \mathbb{E}_2$.*
4. *FEXP$^+$ is the closure of I under composition, limited 3-comprehension and limited doubling recursion.*

Proof (sketch). The first point is essentially as in Cobham [4].

FPSPACE$^+$ is contained in the closure of FP$^+$∪P$^\circ$ by limited recursion on $\mathbb{E}_3$, composition and limited 3-comprehension: First, a superstring-valued FP$^+$∪P$^\circ$ function can compute from the input of a PSPACE Turing machine the transition function of the machine as a table listing the next configuration for each given configuration. Another function in FP$^+$∪P$^\circ$ can compose such a function with itself by reading two (polynomial-sized) entries from this table. Therefore after applying limited recursion on these two functions we obtain a third that outputs the 2^x-step transition function and from this it is trivial to extract the value of the original PSPACE function. Conversely, FPSPACE$^+$ is closed under limited recursion (as each such operation increases the space requirements of a function by a polynomial factor) and the other operations.

For point 3, The step function of a PSPACE Turing machine (a polynomial-time string function) can be iterated exponentially many times using limited doubling recursion restricted to $\mathbb{E}_1 \cup \mathbb{E}_2$. Conversely FPSPACE$^+$ is closed under this restriction of limited doubling recursion as the recursion can be unwound with only a polynomial amount of additional space. This characterization is analogous to the one used in Dowd [8]: initial functions closed under limited recursion. Limited recursion in the context of number functions is of exponential length, as is limited doubling recursion in our setting. This in turn is reminiscent of $\mathcal{E}^2$, the second level of the Grzegorczyk hierarchy [9], which is defined similarly except with an initial function of linear growth rate as opposed to $x\#'y$; this was shown by Ritchie [11] to equal the linear space functions.

Finally, with limited doubling recursion on $\mathbb{E}_3$, the step function of an exp-time Turing machine can be iterated exponentially many times. See [3] for a previous recursion-theoretic characterization of the exponential-time number functions. □

3 Third-Order Bounded Arithmetic Theories

Our main theories are W^i_1 and TW^i_1, intended to correspond to levels of the exponential-time hierarchy; they are parameterized by the type of induction. These theories are suggested by the RSUV isomorphism and are closely connected to U^i_2 and V^i_2, respectively, although we do not claim an actual isomorphism (but one may hold with the unbounded domain versions of these theories).

For $i \geq 0$, W^i_1 is a theory over $\mathcal{L}^3_A$. The axioms of W^i_1 are B1-B14, L1, L2 and SE of [Cook/Kolokolova], (strict) $\forall^2 \Sigma^{\mathcal{B}}_i$-IND and the comprehension schemes $\Sigma^{\mathcal{B}}_0$-2COMP: $(\exists Y \leq t(\overline{x}, \overline{X}))(\forall z \leq a)[\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, z) \leftrightarrow Y(z)]$ and $\Sigma^{\mathcal{B}}_0$-3COMP:

$(\exists\mathcal{Y})(\forall Z \leq a)[\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, Z) \leftrightarrow \mathcal{Y}(Z)]$, where in each case $\phi \in \Sigma_0^{\mathcal{B}}$ subject to the restriction that neither Y nor $\mathcal{Y}$, as appropriate, occurs free in ϕ.

W_1^1 defined above is slightly different than the version in CSL04 [12]; it includes a string equality symbol and extensionality axiom, but this is a conservative extension. The unusual class of formulas for which we admit induction (a bounded string quantifier followed by a strict $\Sigma_i^{\mathcal{B}}$-formula) is in order for a replacement scheme to be provable; as a result of this scheme, W_1^i ultimately admits full $g\Sigma_i^{\mathcal{B}}$-IND; we omit the details.

Define $\widehat{W_1^i}$ to be the analogous theory with the induction scheme restricted to (strict) $\Sigma_i^{\mathcal{B}}$-formulas. Note that $\widehat{W_1^0} = W_1^0$.

TW_1^i is defined identically as above, but with the following scheme named $\Sigma_i^{\mathcal{B}}$-SIND (string or set induction) in place of $\forall^2\Sigma_i^{\mathcal{B}}$-IND:

$$[\forall X, Y, Z((|Z| = 0 \supset \phi(Z)) \wedge (\phi(X) \wedge S(X, Y) \supset \phi(Y)))] \supset \forall Z\phi(Z)$$

for ϕ $\in$(strict)$\Sigma_i^{\mathcal{B}}$, where $S(X, Y)$ is a $\Sigma_0^{\mathcal{B}}$-formula expressing that Y is the lexicographically next finite set after X. Again, TW_1^i admits (string) induction on the more general class of formulas due to a replacement scheme.

TTW_1^i is yet another theory in this vein, with a yet stronger induction scheme named $\Sigma_i^{\mathcal{B}}$-SSIND ("superstring" induction). Note that since (by design) there is no way to bound a third-order object, the scheme refers to a term t, and restricts its attention to the first 2^t bits of the objects. It is intended that this t be some crucial bound from ϕ. The scheme is:

$$[\forall\mathcal{X}, \mathcal{Y}, \mathcal{Z}((\forall X \leq t \neg\mathcal{Z}(X)) \supset \phi(\mathcal{Z})) \wedge (\phi(\mathcal{X}) \wedge S_3(\mathcal{X}, \mathcal{Y}, t) \supset \phi(\mathcal{Y}))] \supset \forall\mathcal{Z}\phi(\mathcal{Z})$$

for ϕ $\in$(strict)$\Sigma_i^{\mathcal{B}}$, where $S_3(\mathcal{X}, \mathcal{Y}, z)$ is a $\Sigma_0^{\mathcal{B}}$-formula expressing that when considering only the lowest 2^z bits of the superstrings, $\mathcal{Y}$ is lexicographically next after $\mathcal{X}$.

The scheme $\Sigma_0^{\mathcal{B}}$-superstring-recursion is $\exists\mathcal{X}\phi^{\text{rec}}(S, \mathcal{X})$, where $\phi(Y, \mathcal{X}) \in \Sigma_0^{\mathcal{B}}$, and $\phi^{\text{rec}}(x, \mathcal{X}) \equiv \forall Y \leq |S|(L_2(Y, S) \supset (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y})))$. $L_2(X, Y)$ expresses that lexicographically, $X < Y$, while $\mathcal{X}^{<Y}$ is a chop function (i.e., $\mathcal{X}^{<Y}(Z)$ abbreviates the subformula $L_2(Z, Y) \wedge \mathcal{X}(Z)$). ϕ (and therefore also ϕ^{rec}) may have other free variables than the displayed ones, but ϕ must have distinguished string and superstring free variables Y and $\mathcal{X}$. ϕ^{rec} then has $\mathcal{X}$ free as well as a new variable S. This scheme is analogous to that from [2] and follows the presentation from [7].

The scheme $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion is $\exists\mathcal{X}\phi^{\text{hrc}}(S, \mathcal{X})$, where $\phi(Y, \mathcal{X}) \in \Sigma_0^{\mathcal{B}}$, and $\phi^{\text{hrc}}(S, \mathcal{X}) \equiv \forall Y \leq |S|(L_2(Y, S) \supset (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y/2})))$, where $\mathcal{X}^{<Y/2}$ is a chop function returning the first $\frac{Y}{2}$ (as a number) bits of $\mathcal{X}$. ϕ and ϕ^{rec} have the same free-variable conventions and requirements as in the superstring recursion scheme. Then HW_1^0 is the theory W_1^0 with the addition of the $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion scheme.

3.1 Definability in the Theories

The definability of functions in the third-order setting is a generalization of the usual definition, but the case of superstring-valued functions additionally includes a mechanism for explicitly reasoning about only an initial segment of the output. This is necessary as superstrings in the theories are formally unbounded, while in the function calculus they are finite.

The following omnibus theorem summarizes results concerning definable functions in the theories; proofs (and the formal definition of definability) are in [13] and are fairly technical, yet generally straightforward. We comment below on especially interesting or unusual points.

Theorem 3. *1. For $i \geq 1$, the $\Sigma_i^{\mathcal{B}}$-definable functions of W_1^i are precisely $(FPSPACE^{(\Sigma_{i-1}^{exp})^\diamond})^+$.*

2. For $i \geq 1$, the $\Sigma_i^{\mathcal{B}}$-definable functions of TW_1^i are precisely $(FEXP^{(\Sigma_{i-1}^{exp})^\diamond})^+$.

3. The $\Sigma_1^{\mathcal{B}}$-definable functions of TTW_1^0 are precisely $FEXP^+$.

4. The $\Sigma_1^{\mathcal{B}}$-definable functions of both HW_1^0 and $\widehat{W_1^1}$ are precisely $FPSPACE^+$.

5. $W_1^0 = TW_1^0$ and the definable functions of this theory are $FPH^+ \cup FPH^\circ$, the polytime hierarchy functions in the third-order setting.

Remarks. 1. Uniqueness of the function value is important, otherwise the definable **multi**-functions are $(\text{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\text{wit,poly}])^+$; analogously to [1] for U_2^1.

2. Straightforward.
3. Functions are defined using the superstring-recursion scheme.
4. Straightforward.
5. Equality of theories is by "shortening of cuts"; the theories are conservative extensions of the second-order polytime hierarchy theory V by a standard argument.

□

3.2 An Application of the Function Calculus

W_1^1 does not seem to be a "minimal" theory for PSPACE as witnessing the induction seems to be too hard. (Put another way, the $\Sigma_2^{\mathcal{B}}$-definable functions of W_1^1 might not be contained in FPSPACE$^+$, analogouslt to the situation for, say, S_2^1). An application of the recursion-theoretic characterization previously presented is that one can specify a language $\mathcal{L}_{PS}$ of FPSPACE$^+$ function symbols with open defining equations. There are cases for initial functions and definitions by limited recursion, as well as minimization functions allowing elimination of quantifiers. The resulting theory, $\overline{HW_1^0}$, is universal; it extends HW_1^0 as it contains functions witnessing the halfrecursion scheme; and the extension is conservative. This last point is proved by showing inductively that each function of $\mathcal{L}_{PS}$ is definable in HW_1^0 by a single application of the halfrecursion operation followed by a $\Sigma_0^{\mathcal{B}}$ projection (a subclass of $\Sigma_1^{\mathcal{B}}$-definable). The upshot of all of this is that HW_1^0 is therefore in some sense a minimal theory for FPSPACE$^+$.

4 Further Research

Some particular problems: First, does $\widehat{W_1^1}$ prove the general induction of W_1^1? One approach is the method of [6], namely by KPT witnessing, with HW_1^0 as the starting point. Second, what about propositional translations of these theories? Partial progress is made in [13]. Third, are any of the theories HW_1^0, W_1^1 or TW_1^1 finitely axiomatizable? Finally, would the conservativity of W_1^1 over HW_1^0 have any complexity-theoretic consequences?

5 Acknowledgment

Thanks to Toniann Pitassi, Charles Rackoff, Alasdair Urquhart, Sam Buss and Stephen Cook for improvements to the presentation and helpful comments; and to the referees, whose detailed suggestions I have but imperfectly heeded.

References

[1] Samuel Buss, Jan Krajíček, and Gaisi Takeuti. On provably total functions in bounded arithmetic theories R_3^i, U_2^i and V_2^i. In Peter Clote and Jan Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 116–61. Oxford University Press, Oxford, 1993.

[2] Samuel R. Buss. Axiomatizations and conservation results for fragments of bounded arithmetic. In *CMWLC: Logic and Computation: Proceedings of a Workshop held at Carnegie Mellon University*, pages 57–84. Contemporary Mathematics Volume 106, American Mathematical Society, 1990.

[3] Peter Clote. A safe recursion scheme for exponential time. In Sergei I. Adian and Anil Nerode, editors, *LFCS97*, volume 1234 of *Lecture Notes in Computer Science*, pages 44–52. Springer, 1997.

[4] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, *Proceedings of the International Congress for Logic, Methodology and Philosophy of Science*, pages 24–30. North-Holland, 1964.

[5] S. A. Cook. CSC 2429S: Proof Complexity and Bounded Arithmetic. Course notes, URL: "http://www.cs.toronto.edu/∼sacook/csc2429h", Winter 2002.

[6] Stephen Cook and Neil Thapen. The strength of replacement in weak arithmetic. In Harald Ganzinger, editor, *LICS04*, pages 256–264. IEEE Computer Society, July 2004.

[7] Stephen A. Cook. Theories for complexity classes and their propositional translations. In Jan Krajíček, editor, *Complexity of Computations and Proofs*, volume 13 of *Quaderni di Matematica*, pages 175–227. Seconda Università di Napoli, 2004.

[8] Martin Dowd. *Propositional Representation of Arithmetic Proofs.* PhD thesis, University of Toronto, 1979.

[9] A. Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematyczne*, 4:1–46, 1953.

[10] Juris Hartmanis. The collapsing hierarchies. *Bulletin of the EATCS*, 33, September 1987.

[11] R. W. Ritchie. Classes of predictably computable functions. *Transactions of the American Mathematical Society*, 106:139–173, 1963.

[12] Alan Skelley. A third-order bounded arithmetic theory for PSPACE. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *CSL04*, volume 3210 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2004.

[13] Alan Skelley. *Theories and Proof Systems for PSPACE and the EXP-Time Hierarchy.* PhD thesis, University of Toronto, 2005. Available from ECCC in the 'theses' section.