On Parallel Hierarchies and R_k^i

Stephen Bloch *
Math/CS Dept. TR 1996–004

September 23, 1996

Abstract

This paper defines natural hierarchies of function and relation classes, constructed from parallel complexity classes in a manner analogous to the polynomial-time hierarchy. A number of structural results about these classes are proven: relationships between them and the levels of PH; relationships between these classes and definability in the bounded arithmetic theories R_k^i; and several results relating conservation among theories of bounded arithmetic to the collapse of complexity classes.

1 Introduction

Rohit Parikh, in [Par71], studied weak theories of arithmetic that have induction only for bounded, or Δ_0, formulæ. These theories became known as *theories of bounded arithmetic*, and were further developed in [Bus86a]. In that paper, Sam Buss restricted induction not only to bounded formulæ, but to bounded formulæ of fixed complexity Σ_i^b. Furthermore, Buss distinguished between an exponential-length and a polynomial-length form of induction, defining thereby two related hierarchies of logical theories T_2^i and S_2^i, and proved basic relationships among them such as that $S_2^i \subseteq T_2^i \subseteq S_2^{i+1}$ for $i \geq 1$. Most importantly, Buss showed that the proof strength of S_2^i corresponded exactly to accepted notions of computational complexity: the functions at level i of the polynomial time hierarchy are precisely those with a Σ_i^b graph, provably total and single-valued in S_2^i.

Expanding on this work, various researchers [All91, Clo89, CT92] introduced a third hierarchy of theories R_2^i based on log-length induction, and demonstrated that R_2^1 analogously characterized the functions computable in the parallel class $\mathcal{NC}$. But no natural description was known of the computational content of R_2^i proofs for $i > 1$, in part because many natural arguments about log-length induction seem to require quasipolynomial, rather than polynomial, growth rates (although see [Blo92, Blo97] for an exception). Attention therefore

*Math/Computer Science Dept, Adelphi University. This work was supported by Judy Goldsmith's NSERC operating grant OGP0121527 while the author was at the University of Manitoba; further work was done variously at the University of Kentucky, Adelphi University, and the DIMACS research center at Rutgers University.

turned to the theories R_3^i, in which terms have quasipolynomial growth. Buss, Krajíček, and Takeuti [BKT93] described the problems for which R_3^i can prove the existence of solutions, relying on a somewhat complex notion of multi-valued functions computed with a limited number of queries to a witnessing oracle.

A preliminary version [Blo95] of the present paper claimed to characterize the functions provably total and *single-valued* in R_2^i, R_3^i, R_4^i, *etc.* by natural oracle-based parallel complexity classes $\square_{i,k}^c$ (defined below; for example, $\square_{1,2}^c = ALOGTIME$)[1], without the complications of multi-valued functions, witnessing oracles, or artificial bounds on the number of oracle queries. S. Buss pointed out a subtle flaw: one of the proofs implicitly relied on the following.

Conjecture 1.1 (S. Buss) *If $R_k^i \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$, with $A \in \Sigma_{i,k}^b$, then there is a $\Sigma_{i,k}^b$ formula B such that $R_k^i \vdash (\forall\vec{x})(\exists! y)B(\vec{x},y)$ and $R_k^i \vdash B(\vec{x},y) \supset A(\vec{x},y)$.*

The corresponding statement for theories S_2^i was shown in [Bus86a], using the polynomial-time self-reducibility of SAT (*i.e.* given a yes/no oracle for SAT, one can *find* a satisfying assignment in polynomial time). Since SAT is not known to be self-reducible in $\square_{1,k}^c$, this technique doesn't carry over easily, and neither [BKT93] nor the author have been able to prove the conjecture for R_k^i. As a result, it is conceivable that some of the $\forall\exists\Sigma_i^b$ consequences of R_k^i correspond to multi-valued functions that R_k^i cannot prove are extensions of $\square_{i,k}^c$ functions.

For another view of the problem, consider the "existential instantiation" inference in a natural deduction system (or its analogues in other styles of proof systems), which given the hypothesis $\exists x\varphi(x)$, introduces a new constant symbol a about which nothing is known except $\varphi(a)$, and reasons with a henceforth. The computational analogue is, given a list of numbers of which at least one has a desired property, to choose an arbitrary number from the list with the desired property. Although polynomial time can do this in many circumstances, it is not clear that $\square_{1,k}^c$ can.

However, we show that $\square_{i,k}^c$ comprises, if not *all* of the $\Sigma_{i,k}^b$ consequences of R_k^i, at least a "large" subset of them: indeed, if we restrict our attention to single-valued functions, $\square_{i,k}^c$ *does* comprise exactly the $\Sigma_{i,k}^b$ consequences of R_k^i. This suffices to prove other results relating conservativity among theories of bounded arithmetic to collapses of complexity classes.

For example, [BKT93] asked whether the known $\forall\exists\Sigma_{i+1}^b$ conservativity between S_3^i and R_3^{i+1} also holds between S_2^i and R_2^{i+1}; we give circumstantial evidence that it does *not*, as such conservativity would imply a collapse of complexity classes (see Theorem 6.11).

With the aid of an ingenious interactive computational model, [KPT91] showed that if $S_2^{i+1} \equiv T_2^i$, then $\Sigma_{i+1}^p \subseteq \Delta_{i+1}^p$/poly and therefore the polynomial hierarchy collapses to at most $\Sigma_{i+2}^p = \Pi_{i+2}^p$. [Bus94] simplified the proof and strengthened the result: under the same hypotheses, $\Sigma_{i+1}^p \subseteq \Delta_{i+1}^p$/poly, and therefore the polynomial hierarchy collapses, *provably in* T_2^i. The present paper shows (see Theorem 7.4) that any equivalence of the form $R_k^{i+1} \equiv S_k^i$, with $i \geq 1, k \geq 3$, would imply a similar collapse of complexity classes, *e.g.* (for $k = 3$) the collapse of the quasipolynomial hierarchy, provably in S_k^i.

[1] The $\square_{i,k}^c$ characterizations depend both on the function-theoretic characterization of [Blo94] and on circuits with oracle gates, as in [Wil85, Wil87].

Section 2 defines the hierarchies of logical theories, the models of parallel computation, and the hierarchies of parallel complexity classes used in this paper. Section 3 proves a variety of basic relationships among these complexity classes. Section 4 shows that R^i_k can define a number of useful functions (*e.g.* sequence coding) and prove a number of useful properties of them. Section 5 shows a direct link between the complexity classes of the present paper and those defined in [BKT93, Kra93]. Section 6 proves that the computational class $\square^c_{i,k}$ is contained in the Σ^b_i consequences of R^i_k. Section 7 gives results analogous to those of [KPT91, Bus94] relating the collapse of theories of bounded arithmetic to the collapse of complexity classes. We conclude by discussing some of the remaining results we would most like to prove about these theories.

2 Definitions

2.1 Universe(s) of Discourse

Most of the existing literature on theories of bounded arithmetic treats the theories as operating over a universe of natural numbers. However, many aspects of parallel computation are easier to formulate in terms of bit-strings. Dyadic notation provides a convenient bijection between the two. To avoid confusion, we shall think of parallel computation as operating on strings in the language $\{\texttt{T}, \texttt{F}\}^*$; when interpreting such a string as a natural number, we read T as the dyadic digit 2 and F as the dyadic digit 1.

2.2 Languages, Notation, and Growth Rates

We work with several first-order languages of arithmetic, similar to those used in [Bus86a] and [BKT93], with whose notation I assume the reader is familiar. First we define a collection of term and relation symbols.

Definition 2.1 *The language $\mathcal{L}_1$ consists of*

- *the variable symbols a, b, c, d, w, x, y, z, possibly subscripted with an integer*
- *the constant symbols 0 and 1, with their usual interpretations*
- *the binary infix relation symbols $=$ and $\leq$ with their usual interpretations*
- *the binary infix function symbols $+$, $\dot{-}$, and $\cdot$ with their usual interpretations (where $x \dot{-} y = \max(0, x - y)$)*
- *the unary "outfix" function symbol $||$ interpreted to mean $\lambda x.\lfloor \log_2(x+1) \rfloor$, the number of digits in x's dyadic representation, and*
- *the binary prefix function symbol* Msp *interpreted to mean $\lambda x.\lfloor \frac{x+1}{2^y} \rfloor \dot{-} 1$ (*i.e. *the result of deleting the rightmost y digits from x's dyadic representation).*

and any term constructed from these by composition and λ-abstraction.

Definition 2.2 *For $k > 1$, the language $\mathcal{L}_k$ is the language $\mathcal{L}_1$ augmented with the binary infix function symbols $\#_2, \#_3, \#_4, \ldots \#_k$, pronounced "smash two", "smash three", and so on. "Smash two" is often simply called "smash".*

The intended interpretations of these symbols are as follows: $x \#_2 y = 2^{|x| \cdot |y|} - 1$, $x \#_3 y = 2^{|x| \#_2 |y|} - 1, \ldots, x \#_k y = 2^{|x| \#_{k-1} |y|} - 1$. These smash functions are discussed in [Cob65, Hoo83, Nel86, Bus86a, All91], all of which use unimportantly different definitions (*e.g.*, using binary, rather than dyadic, lengths, or omitting the "-1").

We shall frequently refer to "the length of an $\mathcal{L}_k$-term", or $|\mathcal{L}_k|$ for short. For example, if we say "$f(x) \in |\mathcal{L}_1|$" or "$f(x)$ is bounded by the length of an $\mathcal{L}_1$ term", we mean that f is at most a linear function in the length of x. Similarly, $|\mathcal{L}_2|$ means a polynomial, or $2^{O(\log)}$, in the lengths of the free variables; $|\mathcal{L}_3|$ is quasipolynomial, or $2^{2^{O(\log\log)}}$, in the lengths of the free variables, and so on. We shall likewise refer to "the log of the length of an $\mathcal{L}_k$-term", abbreviating it $||\mathcal{L}_k||$ (the slight abuse of notation, equating log with length, doesn't affect the results).

One can define a number of other useful functions and relations as abbreviations for terms and formulæ over this basic syntax:

$$
\begin{aligned}
x < y &\Leftrightarrow x + 1 \leq y \\
x \geq y &\Leftrightarrow y \leq x \\
x > y &\Leftrightarrow y < x \\
2 &= 1 + 1 \\
\mathsf{PwrTwo}(x) &\Leftrightarrow |2 \cdot x \dot{-} 2| + 1 = |2 \cdot x \dot{-} 1| \\
x^2 &= x \cdot x \\
2^{|x|} &= (x \#_2 1) + 1 \\
\mathsf{Lsp}(x, y) &= x \dot{-} 2^{\min(y, |x|)} \cdot \mathsf{Msp}(x, y) \\
\mathsf{Substr}(x, y, z) &= \mathsf{Lsp}(\mathsf{Msp}(x, y), z) \\
\mathsf{Bit}(i, x) &= \mathsf{Substr}(x, i, 1) \\
\left\lfloor \frac{x}{2} \right\rfloor &= \mathsf{Msp}(x + 1, 1) \\
\mathsf{Parity}(x) &= \mathsf{Lsp}(x, 1) \\
\mathsf{Conc}(x, y) &= x \cdot 2^{|y|} + y \\
\mathsf{Shl}(x, y) &= \mathsf{Lsp}(\mathsf{Conc}(x, y), |x|) \\
\mathsf{Fh}(x) &= \mathsf{Msp}\left(x, \left\lfloor \frac{|x|}{2} \right\rfloor\right) \\
\mathsf{Bh}(x) &= \mathsf{Lsp}\left(x, \left\lfloor \frac{|x|}{2} \right\rfloor\right)
\end{aligned}
$$

Note that the Conc, Msp, and Lsp functions act as inverses: $\mathsf{Msp}(\mathsf{Conc}(x, y), |y|) = x$, $\mathsf{Lsp}(\mathsf{Conc}(x, y), |y|) = y$, and (for $(|y| \leq |x|)$ $\mathsf{Conc}(\mathsf{Msp}(x, y), \mathsf{Lsp}(x, y)) = x$. (Allen, in [All91], defined these functions differently, operating on binary rather than dyadic numbers, with the odd result that the "back half" of a 40-digit number could be anywhere from 0 to 20 digits long, depending on leading zeroes; in our setting it will always be exactly 20 digits.)

2.3 Syntactic Complexity

We next define an hierarchy of syntactic formula classes, analogous to the classes Σ_i, Π_i, Δ_i familiar to recursion theorists. The following definitions differ from those in [Bus86a] mainly in the presence of a second subscript k; since Buss was only concerned with a language analogous to $\mathcal{L}_2$, his results apply to our classes with $k = 2$.

Definition 2.3 *A bounded quantifier with bound of the form* $|t(x)|$, i.e. *one whose outermost function symbol is* $|\cdots|$, *is said to be* sharply bounded.

Definition 2.4 *The classes $\Sigma^b_{i,k}$ and $\Pi^b_{i,k}$ are the smallest sets of formulæ satisfying the following inductive definition (where t is a term in language $\mathcal{L}_k$).*

- $\Sigma^b_{0,k} = \Pi^b_{0,k}$ *is the set of formulæ in language $\mathcal{L}_k$ in which all quantifiers are sharply bounded.*
- $\Pi^b_{i,k} \subseteq \Sigma^b_{i+1,k}$.
- $\Sigma^b_{i,k} \subseteq \Pi^b_{i+1,k}$.
- *If $A \in \Sigma^b_{i+1,k}$, then the formula $(\exists x \le t)A$ is also in $\Sigma^b_{i+1,k}$.*
- *If $A \in \Pi^b_{i+1,k}$, then the formula $(\forall x \le t)A$ is also in $\Pi^b_{i+1,k}$.*
- *If $A, B \in \Sigma^b_{i,k}$ (respectively $\Pi^b_{i,k}$), then so are $A \wedge B$ and $A \vee B$.*
- *If $A \in \Sigma^b_{i,k}$ and $B \in \Pi^b_{i,k}$ (respectively* vice versa*), then $(A \supset B) \in \Pi^b_{i,k}$ (respectively $\Sigma^b_{i,k}$).*
- *If $A \in \Sigma^b_{i,k}$ (respectively $\Pi^b_{i,k}$), then so are both $(\forall x \le |t|)A$ and $(\exists x \le |t|)A$.*

When the language $\mathcal{L}_k$ is clear from context, particularly when $k = 2$, we omit the subscript k.

2.4 Classical Theories of Bounded Arithmetic

The theory T^i_k is axiomatized by a fixed, finite set of quantifier-free axioms and the inference rule $\Sigma^b_{i,k} - IND$, which we write in Gentzen sequent-calculus form:

$$\frac{\Gamma(\vec{b}), A(a, \vec{b}) \longrightarrow A(a+1, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}) \longrightarrow A(t(\vec{b}), \vec{b}), \Delta(\vec{b})}$$

where $A \in \Sigma^b_{i,k}$ and $t \in \mathcal{L}_k$.

The theory S^i_k is axiomatized similarly, but with one of two (provably equivalent; see [Bus86a, All91]) induction inferences in place of IND.

$\Sigma^b_{i,k} - PIND$ ("prefix induction"):

$$\frac{\Gamma(\vec{b}), A\left(\lfloor \frac{a}{2} \rfloor, \vec{b}\right) \longrightarrow A(a, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}) \longrightarrow A(t(\vec{b}), \vec{b}), \Delta(\vec{b})}$$

$\Sigma^b_{i,k} - LIND$ ("length induction"):

$$\frac{\Gamma(\vec{b}), A(a, \vec{b}) \longrightarrow A(a+1, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}) \longrightarrow A(|t(\vec{b})|, \vec{b}), \Delta(\vec{b})}$$

The theory R^i_k is axiomatized similarly, with one of four induction schemes, called DCI, $PPIND$, $LPIND$, and $LLIND$. They are shown to be equivalent in [All91, CT92] (in a sufficiently expressive language, such as $\mathcal{L}_k$ for $k \geq 2$).

The DCI induction scheme introduced in [All91] uses functions Fh and Bh that return the "front half" and "back half" respectively of a number's binary representation. We use, instead, the more well-behaved Fh and Bh functions defined in section 2.2. As a side effect, we must use more base cases: in addition to 0 and 1, we need 2 in order for DCI to be valid.

$\Sigma^b_{i,k} - DCI$ ("divide and conquer induction"):

$$\frac{\Gamma(\vec{b}), A(\mathsf{Bh}(a), \vec{b}), A(\mathsf{Fh}(a), \vec{b}) \longrightarrow A(a, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}), A(1, \vec{b}), A(2, \vec{b}) \longrightarrow A(t(\vec{b}), \vec{b}), \Delta(\vec{b})}$$

$\Sigma^b_{i,k} - PPIND$:

$$\frac{\Gamma(\vec{b}), A(\mathsf{Fh}(a), \vec{b}) \longrightarrow A(a, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(1, \vec{b}), A(2, \vec{b}) \longrightarrow A(t(\vec{b}), \vec{b}), \Delta(\vec{b})}$$

$\Sigma^b_{i,k} - LPIND$:

$$\frac{\Gamma(\vec{b}), A\left(\lfloor \frac{a}{2} \rfloor, \vec{b}\right) \longrightarrow A(a, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}) \longrightarrow A(|t(\vec{b})|, \vec{b}), \Delta(\vec{b})}$$

$\Sigma^b_{i,k} - LLIND$:

$$\frac{\Gamma(\vec{b}), A(a, \vec{b}) \longrightarrow A(a+1, \vec{b}), \Delta(\vec{b})}{\Gamma(\vec{b}), A(0, \vec{b}) \longrightarrow A(||t(\vec{b})||, \vec{b}), \Delta(\vec{b})}$$

The reader seeking more detail and discussion of these theories and induction schemes is referred to [Bus86a, All91, CT92, Blo92].

2.5 Axiomatizing Bounded Arithmetic

The theory R^1_2 seems to require a few more base functions than S^1_2. Where [CU93] used the language $\{0, Sx, |x|, \lfloor \frac{x}{2} \rfloor, x+y, x \cdot y, x\#y\}$ for their theory IS^1_2, [All91] generalizes the $\lfloor \frac{x}{2} \rfloor$ function to $\mathsf{Msp}(x, y)$, and adds the functions $x \dot{-} y$, $\mathsf{Bit}(i, x)$, and $\mathsf{Lsp}(x, i)$ in defining a theory similar to R^1_2. We shall do the same, except that we defined Bit and Lsp by abbreviation (see section 2.2).

The defining axioms for all these symbols follow. Fourteen, labelled "(CU)", are (straightforward adaptations of) corresponding axioms in [CU93]. Several others, labelled (A), appear (modulo obvious adaptations) in [All91]. I use different defining axioms for Msp and # from those in [All91].

- $0 \leq x$ (CU)
- $x \leq 1 \supset x = 0 \vee x = 1$ (A)
- $x \leq y \supset (x = y \vee x + 1 \leq y)$ (CU)

- $(x \le y \land y \le z) \supset x \le z$ (CU)
- $(x \le y \land y \le x) \supset x = y$ (CU)
- $x \le y \lor y \le x$ (CU)

- $|0| = 0$ (CU)
- $|(2 \cdot x) + 1| = |x| + 1$ (CU)
- $|(2 \cdot x) + 2| = |x| + 1$ (CU)
- $x \le y \supset |x| \le |y|$ (CU)

- $x + 0 = x$ (CU)
- $x + y = y + x$ (A)
- $(x + y) + z = x + (y + z)$ (CU)
- $x + y \le x + z \leftrightarrow y \le z$ (CU)
- $x \le y \supset x \dot{-} y = 0$ (A)
- $y \le x \supset (x \dot{-} y) + y = x$ (A)

- $x \cdot 1 = x$ (CU)
- $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ (CU)

- $\mathsf{Msp}(x, 0) = x$ (A)
- $|x| \le i \supset \mathsf{Msp}(x, i) = 0$ (A)
- $(\mathsf{Msp}(x, i) = 2 \cdot \mathsf{Msp}(x, i + 1) + 1) \lor (\mathsf{Msp}(x, i) = 2 \cdot \mathsf{Msp}(x, i + 1) + 2)$ (A)
- $\mathsf{Msp}(2 \cdot x + 1, i + 1) = \mathsf{Msp}(x, i)$
- $\mathsf{Msp}(2 \cdot x + 2, i + 1) = \mathsf{Msp}(2 \cdot x + 1, i + 1)$

The present paper defines the smash functions slightly differently than does [CU93], as mentioned in section 2.2, so the axioms for $\#_2, \ldots, \#_k$ become simply

- $|x \#_2 y| = |x| \cdot |y|$
- $|(x \#_2 y) \dot{-} 1| = |x \#_2 y| \dot{-} 1$

- $\vdots$
- $|x\#_k y| = |x|\#_{k-1}|y|$
- $|(x\#_k y) \dot{-} 1| = |x\#_k y| \dot{-} 1$

In addition, the theory R^i_k has the inference rules of LKB (see [Bus86a]) and *DCI* on all formulæ in $\Sigma^b_{i,k}$.

2.6 Circuits and Uniformity

Our complexity classes are defined in terms of uniform families of multiple-output circuits, which furthermore may contain "oracle gates". (In addition, for technical reasons, we introduce "output gates" which compute the identity function of their single input.) Wilson [Wil85, Wil87] introduced a similar notion of oracle gates, and I follow his convention that an oracle gate's "size" and "depth" are its fanin and the log of its fanin respectively. (This convention is crucial to some of the proofs in the $k = 2$ case; it makes no difference for $k \geq 3$.)

Most of the circuit families in this paper are described by *genus*:

Definition 2.5 *An oracle circuit family of* genus k *is a circuit family with at most* $|\mathcal{L}_k|$ *output gates, depth at most* $||\mathcal{L}_k||$*, and fanin at most 2 except for its oracle gates. All oracle gates in a given circuit family with input* $\vec{x}$ *have fanin exactly* $|t(\vec{x})|$*, where* $t \in \mathcal{L}_k$ *is a nontrivial term in the following sense.*

Definition 2.6 *A term* $t(\vec{x})$ *is* nontrivial *if it is bounded below by the maximum of its free variables.*

Actually, we only need that $|t(\vec{x})| = \Omega(|\vec{x}|)$, *i.e.*, t is bounded below by a polynomial in which all its free variables $\vec{x}$ appear with positive degree and positive coefficient. But for our purposes, this more inclusive definition does not seem worth its added complexity.

Most results in this paper are stated "for $k \geq 2$" or "for $k \geq 3$". Some would hold even for $k = 1$, but genus 1 circuit families are so ill-behaved (*e.g.*, not closed under composition) that we shall usually ignore the $k = 1$ case.

Lemma 2.7 *For* $k \geq 2$*, an oracle circuit family of genus* k *has size (*i.e.*, number of gates) at most* $|\mathcal{L}_k|$*.*

PROOF: Recall that by convention, an oracle gate has the same "size" as a tree of bounded-fanin gates of the same "depth". So for purposes of size, we may assume there are no oracle gates. Thus the size of any *single*-output circuit is $|\mathcal{L}_k|$. There may be as many as $|\mathcal{L}_k|$ output gates in a circuit. But since $|\mathcal{L}_k|$ is closed under multiplication for $k \geq 2$, the total size remains $|\mathcal{L}_k|$. ∎

All circuit families in this paper are assumed to be uniform in the U_{E^*} sense of [Ruz81]. However, we must slightly extend Ruzzo's definition of the extended connection language to handle multiple-output circuits and oracle gates. The following definition is formulated using strings of 0's and 1's interpreted in binary; however, this notation can be easily interconverted with dyadic by using $+$, $\dot{-}$, and $2^{|x|}$.

Definition 2.8 *The* descendant *of a gate γ along path $p \in \{0,1\}^*$ is defined by the following algorithm:*

> *let "the current gate" be γ*
> *while $|p| > 0$,*
>   *let r be the fanin of the current gate (either 0, 1, 2, or $|t|$)*
>   *(if $r = 0$, the descendant is undefined)*
>   *let c be the binary number represented by the leftmost* $\max(1, \lceil \log(r) \rceil)$ *bits of p*
>   *(if $|p| < \lceil \log(r) \rceil$, the descendant is undefined)*
>   *remove the leftmost $\lceil \log(r) \rceil$ bits of p*
>   *update the current gate to be the c-th leftmost child of the current gate*
> *return the current gate*

Definition 2.9 *Each gate in an oracle circuit has a* type, *encoded as a binary number, as follows:*

- *The type of an input gate is an ordered pair: a constant-length indication that it* is *an input gate, and a binary number indicating which bit of input it is.*
- *The type of an output gate is an ordered pair: a constant-length indication that it* is *an output gate, and a binary number indicating which bit of output it is. (An output gate is assumed to have fanin exactly 1, and to compute the identity function.)*
- *The type of a bounded-fanin internal gate is an ordered pair: a constant-length indication that it* is *a bounded internal gate, and a constant-length indication of what Boolean function it computes.*
- *The type of an oracle gate for a relation ρ is a constant-length indication that it* is *a ρ-oracle gate.*

Definition 2.10 *The* extended connection language *of an oracle circuit family is the set of 4-tuples $\langle n, \gamma, p, \tau \rangle$ such that n is the binary representation of the number of inputs to the circuit, γ is a binary number identifying a gate in the circuit, $p \in \{0,1\}^*$, and either $|p| = 0$ and τ is the type of gate γ, or $|p| \geq 1$ and τ is a binary number identifying the descendant of γ along path p.*

Definition 2.11 (Ruzzo) *A family of circuits of depth $T(n)$ and size $Z(n)$ is U_{E^*}-uniform if its extended connection language is recognizable by an alternating Turing machine (with random access to a read-only input tape) in time $O(T(n))$ and space $O(\log(Z(n)))$.*

Lemma 2.12 *An oracle-free circuit family of genus $k \geq 2$ is U_{E^*}-uniform iff its extended connection language is in $ATIME(||\mathcal{L}_k||)$.*

PROOF: Straightforward; note that $k \geq 2$ is necessary in order for the depth bounds $||\mathcal{L}_k||$ to be closed under constant factors. ∎

The class of such circuit families is "self-uniform" in that they are exactly as powerful as the ATM's that define their uniformity (see Theorems 3 and 4 of [Ruz81]). Henceforth whenever we speak of "uniform" circuit families, we mean this sense of uniformity.

2.7 Complexity Classes

The polynomial-time hierarchy contains relation classes Σ_i^p, Π_i^p, and Δ_i^p, and function classes $\Box_i^p$ (see, *e.g.*, [Bus86a]). Buss mentions in passing that analogous definitions and theorems hold if the language is expanded to $\mathcal{L}_k$, for $k \geq 2$. We therefore generalize the notation:

Definition 2.13 *The complexity classes $\Sigma_{i,k}^p$, $\Pi_{i,k}^p$, $\Delta_{i,k}^p$, and $\Box_{i,k}^p$, for $k \geq 1$, are defined as follows:*

- $\Sigma_{0,k}^p = \Pi_{0,k}^p = \Delta_{0,k}^p = DTIME(|\mathcal{L}_k|)$.
- *For $i \geq 0$, $\Sigma_{i+1,k}^p$ is the closure of $\Pi_{i,k}^p$ under $\mathcal{L}_k$-bounded existential quantification.*
- *For $i \geq 0$, $\Pi_{i+1,k}^p$ is the closure of $\Sigma_{i,k}^p$ under $\mathcal{L}_k$-bounded universal quantification.*
- *For $i \geq 0$, $\Delta_{i+1,k}^p = DTIME(|\mathcal{L}_k|)^{\Sigma_{i,k}^p} = DTIME(|\mathcal{L}_k|)^{\Pi_{i,k}^p}$.*
- $\Box_{0,k}^p = FDTIME(|\mathcal{L}_k|)$, i.e., *the functions computable in $DTIME(|\mathcal{L}_k|)$.*
- *For $i \geq 0$, $\Box_{i+1,k}^p = FDTIME(|\mathcal{L}_k|)^{\Sigma_{i,k}^p} = FDTIME(|\mathcal{L}_k|)^{\Pi_{i,k}^p}$.*

For example, $\Delta_{1,2}^p$ is $\mathcal{P}$, $\Sigma_{1,2}^p$ is $\mathcal{NP}$, and $\Box_{2,3}^p$ is the class of functions computable in quasipolynomial time with oracles for nondeterministic quasipolynomial time. We also define analogous classes based on *parallel* complexity:

Definition 2.14 *The complexity classes $\Sigma_{i,k}^c$, $\Pi_{i,k}^c$, $\Delta_{i,k}^c$, and $\Box_{i,k}^c$, for $k \geq 1$, are defined as follows:*

- *$\Sigma_{0,k}^c = \Pi_{0,k}^c = \Delta_{0,k}^c =$ the class of relations decidable by a uniform family of single-output circuits of genus k, containing no oracle gates.*
- *For $i \geq 0$, $\Sigma_{i+1,k}^c$ is the closure of $\Pi_{i,k}^c$ under $\mathcal{L}_k$-bounded existential quantification*
- *For $i \geq 0$, $\Pi_{i+1,k}^c$ is the closure of $\Sigma_{i,k}^c$ under $\mathcal{L}_k$-bounded universal quantification.*
- *For $i \geq 0$, $\Delta_{i+1,k}^c$ is the class of relations decidable by a uniform family of single-output circuits of genus k, containing oracle gates for some $\Sigma_{i,k}^c$ or $\Pi_{i,k}^c$ relation.*
- *$\Box_{0,k}^c$ is the class of functions computable by a uniform family of $|\mathcal{L}_k|$-output circuits of genus k, with no oracle gates.*

- *For $i \geq 0$, $\Box^c_{i+1,k}$ is the class of functions computable by a uniform family of $|\mathcal{L}_k|$-output circuits of genus k, containing oracle gates for some $\Sigma^c_{i,k}$ or $\Pi^c_{i,k}$ relation.*

Henceforth families with the uniformity, size, depth, and gate restrictions above will be called $\Sigma^c_{i,k}$, $\Pi^c_{i,k}$, and $\Delta^c_{i,k}$ circuit families respectively.

At the $k = 2$ level, it is not clear whether the "most natural" parallel complexity class is $\mathcal{NC}^1$ or $\mathcal{NC}$. For example, Corollary 5.8 below characterizes $\Box^c_{i,k}$ (which for $k = 2$ is based on $\mathcal{NC}^1$). But although Theorem 6.1 also holds for $\Box^c_{i,k}$, it can be strengthened in the $k = 2$ case to apply to classes based on $\mathcal{NC}$. We now define these classes precisely.

Definition 2.15 (Allen) *$\mathcal{FNC}$ is the class of functions computable by U_{E^*}-uniform[2] circuit families with polylog depth, polynomial size, and polynomially many outputs.*

Definition 2.16 *For any class X of relations on $\{0,1\}^*$,*

- *$\mathcal{FNC}^X$ is the class of functions computable by U_{E^*}-uniform, polylog depth, polynomial size, oracle circuit families with oracle gates of fanin $|t|$ (where $t \in \mathcal{L}_2$ is nontrivial) for an X relation.*
- *$\mathcal{NC}^X$ is the class of relations whose characteristic functions are 0/1-valued $\mathcal{FNC}^X$ functions.*

As usual, an oracle gate's "depth" and "size" are considered to be $||t||$ and $|t|$ respectively. Note that since t is nontrivial, these are $\theta(||\mathcal{L}_k||)$ and $\theta(|\mathcal{L}_k|)$ respectively.

Normally we shall study classes of the form $\mathcal{NC}^{\Sigma^c_{i-1,2}}$ and $\mathcal{FNC}^{\Sigma^c_{i-1,2}}$, by analogy to $\Delta^c_{i,k}$ and $\Box^c_{i,k}$.

2.8 Alternating Turing machines

We can also define an alternating Turing machine model to correspond to arbitrary levels in the Δ^c hierarchies.

Definition 2.17 *An* oracle alternating Turing machine of genus k *is an alternating $||\mathcal{L}_k||$-time Turing machine whose states are partitioned into universal, existential, exclusive-or, and oracle states. The universal, existential, and exclusive-or states each have exactly two next states. The behavior of all oracle states in a given machine depends on a nontrivial global bound $t \in \mathcal{L}_k$.*

When the machine enters an oracle state for relation ρ, it guesses a binary number $0 \leq j < |t(x)|$, writes it and a delimiter at the end of a special oracle tape, and goes on to a next state deterministically. Let D_j denote the accept/reject decision of this next state given the number j; then the oracle state accepts iff the relation ρ holds of the $|t|$-bit string $D_{|t|-1}D_{|t|-2}\cdots D_2D_1D_0$. An oracle state is counted as taking $||t||$ time steps.

[2] Allen actually defined $\mathcal{FNC}$ circuit families to be logspace uniform, but as he points out, it makes no difference at this level;see [Ruz81].

(Recall that [CKS81] defined ATM's with NOT states, then proved that they could be eliminated. This proof appears not to work in the presence of oracle states, unless the oracles come from a class closed under complement. We use two-input exclusive-or states instead, as they can be "programmed" to compute either a NOT or an identity function by making one of their successor configurations unconditionally accept or reject.)

For example, if ρ were a $|t|$-way AND, a ρ-oracle state would be equivalent to universally guessing $||t||$ bits, which could have equally well been done without an oracle in the same time. Similar reasoning holds for any $\rho \in \Delta^c_{0,k}$, giving us

Lemma 2.18 *For $k \geq 2$, a relation is computable by an oracle ATM of genus k with a $\Delta^c_{0,k}$ oracle iff it is computable by an ATM of genus k with no oracles at all.*

A more interesting situation arises when ρ is, say, a $\Sigma^c_{1,k}$ relation; then there is no obvious way to simulate the oracle in less than $|\mathcal{L}_k|$ parallel time, exponentially too long. The following lemma may be seen as a generalization of Ruzzo's theorems 3 and 4.

Lemma 2.19 *For $k \geq 2$, a relation is computable by an ATM of genus k with an oracle in $\Sigma^c_{i-1,k}$ iff the relation is in $\Delta^c_{i,k}$.*

Proof: Let ρ be $\Delta^c_{i,k}$, using gates for $\Sigma^c_{i-1,k}$ oracle σ. We construct an ATM to compute ρ as follows. At all times, maintain on an extra tape the path from the root to the gate currently being simulated. Guess (in constant time) whether the gate is AND, OR, NOT, output, input, or oracle. Depending on the answer, then guess the remaining bits of its type. In any case, you now know the complete type of the gate; confirm this type in parallel with the U_{E^*} uniformity algorithm. If AND or OR, simulate it by making the appropriate guess and appending a bit to the path. If NOT, simulate it with an exclusive-or state one of whose successors accepts unconditionally. If a σ oracle gate, simulate it with a σ oracle state, appending the number guessed to the path. If input, simulate it by accessing the input tape. If output, return an accept/reject decision.

Note that guessing the type of a gate takes constant time except for input and output gates, of which any given path contains at most one each. Thus the total time to simulate a path of length $||\mathcal{L}_k||$ is at most $O(||\mathcal{L}_k||)$, which is still $||\mathcal{L}_k||$ for $k \geq 2$.

Conversely, let ρ be computable by an ATM of genus k with a $\Sigma^c_{i-1,k}$ oracle σ. By standard techniques, assume all access to the input tape is at leaves. Furthermore, assume the ATM never overwrites the numbers it gets from oracle states; if it wishes to overwrite them, it has plenty of time to copy them onto writable work tapes first. As in [Ruz81], we construct a circuit with a gate labelled (t, a) for each time step t within the $||\mathcal{L}_k||$ time bounds and each configuration a within the $||\mathcal{L}_k||$ space bounds, and connect them in the obvious way. A σ-oracle state of the ATM, of course, becomes a σ-oracle gate in the circuit; at any subsequent time that the ATM reads from an oracle tape, the subcircuit reached by taking branch j through the oracle gate acts as though the ATM had read a j from the oracle tape. Note that this circuit family is still U_{E^*}-uniform: given an input length and an instance of the extended connection language, the uniformity condition need only check whether all gates in the instance are labelled with suitable times and configurations, and whether the path in the instance respects the allowable fan-in of gates, and then follow paths by simulating ATM actions. ∎

In particular, $\Delta^c_{0,k}$ is equal to $ATIME(||\mathcal{L}_k||)$ for all $k \geq 2$. For example, $\Delta^c_{0,2}$ is known in the literature as ALOGTIME or uniform $\mathcal{NC}^1$, and $\Delta^c_{0,3}$ is the class of problems solvable in uniform polylog depth, or alternating polylog time, or deterministic polylog space (for other relevant characterizations of these classes, see [Blo94]).

3 Basic Relationships among the Classes

In this section we prove "the easy theorems" about the classes defined in section 2.7, showing they constitute a well-behaved complexity hierarchy, before proceeding to more profound results.

Lemma 3.1 *For $k \geq 2$, $\Delta^c_{0,k} \subseteq \Delta^p_{0,k}$.*

PROOF: A $\Delta^c_{0,k}$ circuit family has size $|\mathcal{L}_k|$, so a deterministic $|\mathcal{L}_k|$-time program can construct the appropriate circuit and evaluate it. ∎

Lemma 3.2 *For $k \geq 2$, $\Sigma^c_{1,k} = \Sigma^p_{1,k}$.*

PROOF: The $\subseteq$ direction follows immediately from Lemma 3.1.

For the $\supseteq$ direction, consider an arbitrary $\Sigma^p_{1,k}$ machine, *i.e.*, a nondeterministic $|\mathcal{L}_k|$-time machine. Its computation can be simulated in $\Sigma^c_{1,k}$ by guessing a string representing a complete computation of the machine, then checking that the first configuration contains the right input and initial conditions, the last configuration is accepting, and (in parallel for all j) that the $j+1$-th configuration follows correctly from the j-th. The string is of length $|\mathcal{L}_k|^2$, which for $k \geq 2$ is $|\mathcal{L}_k|$. The parallel check requires a $|\mathcal{L}_k|$-way conjunction, which can be done easily by a $\Delta^c_{0,k}$ circuit family. Checking that a configuration follows correctly from the previous one requires checking that their tape contents are identical except for the square under the head in the earlier configuration; this again is a $|\mathcal{L}_k|$-way conjunction, and easily within $\Delta^c_{0,k}$. ∎

This is not an earthshaking result; similar statements date at least to [Fag74]. But in our notation, it implies immediately the following theorem:

Theorem 3.3 *For $i \geq 1, k \geq 2$, $\Sigma^c_{i,k} = \Sigma^p_{i,k}$ and $\Pi^c_{i,k} = \Pi^p_{i,k}$.*

In light of this theorem, we can dispense with proving basic properties of the $\Sigma^c_{i,k}$ and $\Pi^c_{i,k}$ classes: either we already know them, or we are unlikely to be able to prove them with present techniques. Indeed, we can dispense with the notations $\Sigma^c_{i,k}$ and $\Pi^c_{i,k}$ altogether. But we still don't know much about the $\Delta^c_{i,k}$ and $\Box^c_{i,k}$ classes. So we'll prove a number of useful results about these classes and the relationships among them. The most trivial ones I state without proof:

Fact 3.4

- *The $\Delta^c_{i,k}$ relations are the relations with characteristic functions in $\Box^c_{i,k}$.*
- *For $i \geq 0$, $k \geq 1$, $\Delta^c_{i,k}$ is closed under negation, finite disjunction and conjunction, and definition by $\Delta^c_{i,k}$ cases (and similarly with $\mathcal{NC}^X$ in place of $\Delta^c_{i,k}$).*
- *For all $i' \geq i \geq 0$ and $k' \geq k \geq 1$, we have $\Delta^c_{i,k} \subseteq \Delta^c_{i',k'}$ and $\Box^c_{i,k} \subseteq \Box^c_{i',k'}$.*

- $\Sigma^c_{i,k} \cup \Pi^c_{i,k} \subseteq \Delta^c_{i+1,k}$.

Lemma 3.5 *For $i \geq 0$ and $k \geq 2$, $\square^c_{i,k}$ and $\mathcal{FNC}^X$ are closed under composition.*

PROOF: Suppose functions $f(y)$ and $g(x)$ are in $\square^c_{i,k}$. A multiple-output oracle circuit for f can be converted into one for $f \circ g$ by replacing the input gates for bits of y with the outputs of a multiple-output circuit for g. The resulting circuit is still $\square^c_{i,k}$, because circuit depth is closed under addition for $k \geq 2$.

The proof for $\mathcal{FNC}^X$ is similar. ∎

Corollary 3.6 *For $i \geq 0$ and $k \geq 2$, $\Delta^c_{i,k}$ is closed under substitution by $\square^c_{i,k}$ functions (similarly, $\mathcal{NC}^X$ is closed under $\mathcal{FNC}^X$ substitution).*

Lemma 3.7 *For $k \geq 2$, $\Delta^c_{0,k} = \Delta^c_{1,k}$.* *(Similarly, $\mathcal{NC} = \mathcal{NC}^{\mathcal{NC}}$.)*

PROOF: Let $\rho \in \Delta^c_{1,k}$. Then ρ has a uniform circuit family of genus k containing oracle gates for some $\Sigma^c_{0,k}$ (*i.e.*, $\Delta^c_{0,k}$) relation σ. To compute ρ in $\Delta^c_{0,k}$, simply replace each oracle gate for σ with an entire circuit for σ (depth $||\mathcal{L}_k||$). If $k = 2$, then a σ-oracle gate is counted as having depth $\theta(\log(|x|))$ and the total depth of a ρ-circuit is $O(\log(|x|))$, so each path can have had only constantly many σ-oracle gates. Replacing them with $||\mathcal{L}_k||$-depth σ-circuits adds only $O(||\mathcal{L}_k||)$ to the total depth, so the result has depth $||\mathcal{L}_k||$. On the other hand, if $k \geq 3$, then replacing each σ-oracle gate with a $||\mathcal{L}_k||$-depth circuit at most multiplies the depth by $||\mathcal{L}_k||$, which is still within $||\mathcal{L}_k||$ for $k \geq 3$. The uniformity of the new circuit family follows easily from the uniformity of the ρ family and the σ family.

The proof for $\mathcal{NC}$ is similar, relying on the convention that an oracle gate with fanin $|t|$ is counted as having size $|t|$. ∎

Lemma 3.8 *For any $i > 0$, $k \geq 2$, the function $f(\vec{x})$ is in $\square^c_{i,k}$ iff its* bit-graph*, the relation $\lambda j, \vec{x}.\mathsf{Bit}(j, f(\vec{x}))$, is in $\Delta^c_{i,k}$.*

Similarly, a function is in $\mathcal{FNC}^X$ iff its bit-graph is in $\mathcal{NC}^X$.

PROOF: ($\Rightarrow$): A circuit to do this consists of a $|\mathcal{L}_k|$-bit multiplexer (which can be done in depth $||\mathcal{L}_k||$) composed with a circuit for f.

($\Leftarrow$): Simply use $|f(\vec{x})|$ circuits in parallel, each computing a different bit of f. ∎

Lemma 3.9 *For $i \geq 0$ and $k \geq 2$, if $\rho(\vec{x})$ is equivalent to $(\exists y \leq |s(\vec{x})|)\sigma(\vec{x}, y)$ or to $(\forall y \leq |s(\vec{x})|)\sigma(\vec{x}, y)$, where $\sigma \in \Delta^c_{i,k}$ and $s \in \mathcal{L}_k$, (respectively $\sigma \in \mathcal{NC}^X$, with $s \in \mathcal{L}_2$), then ρ is itself in $\Delta^c_{i,k}$ (respectively $\mathcal{NC}^X$).*

That is, $\Delta^c_{i,k}$ and $\mathcal{NC}^X$ are is closed under sharply-bounded quantifiers.

PROOF: An AND or OR of fanin $|\mathcal{L}_k|$ can be evaluated by a bounded-fanin circuit of depth $||\mathcal{L}_k||$, *i.e.*, at the cost of an additional $||\mathcal{L}_k||$ depth, allowable for $k \geq 2$. ∎

Lemma 3.10 *For $i \geq 0, k \geq 2$, $\Delta^c_{i,k} \subseteq \Delta^p_{i,k}$.* *(Similarly, $\mathcal{NC}^{\Sigma^p_{i-1,2}} \subseteq \Delta^p_{i,2}$.)*

PROOF: The proof is similar to that of Lemma 3.1. A $\Delta^p_{i,k}$ program has time to construct a description of the $\Delta^c_{i,k}$ circuit (complete with oracle gates) and evaluate it, invoking a $\Sigma^p_{i-1,k}$ oracle when necessary. ∎

Corollary 3.11 *For $i \geq 0, k \geq 2$, $\Delta^c_{i,k} \subseteq \Sigma^p_{i,k} \cap \Pi^p_{i,k}$.*
(Similarly, $\mathcal{NC}^{\Sigma^p_{i-1,2}} \subseteq \Sigma^p_{i,2} \cap \Pi^p_{i,2}$.)

Corollary 3.12 *For $i \geq 0$, $k \geq 2$, the classes $\Sigma^p_{i,k}$ and $\Pi^p_{i,k}$ are closed under definition by cases, as long as the decision is in either $\Delta^c_{i,k}$ or $\mathcal{NC}^{\Sigma^p_{i-1,2}}$.*

Lemma 3.13 *For $i \geq 1, k \geq 2$, $\Box^c_{i,k} \subseteq \Box^p_{i,k}$ and $\mathcal{FNC}^{\Sigma^p_{i-1,2}} \subseteq \Box^p_{i,2}$.*

PROOF: Similar to that of Lemma 3.10. A $\Box^p_{i,k}$ program has time to construct, for each of $|\mathcal{L}_k|$ output bits j, a complete description of the $\Delta^c_{i,k}$ circuit that computes the j-th output bit. The $\Box^p_{i,k}$ program can then evaluate each of these circuits and concatenate the answers. All of this works because $|\mathcal{L}_k|$ is closed under multiplication for $k \geq 2$. ∎

Corollary 3.14 *For $i \geq 0$, $k \geq 2$, the classes $\Sigma^p_{i,k}$ and $\Pi^p_{i,k}$ are closed under substitution by $\Box^c_{i,k}$ (and $\mathcal{FNC}^{\Sigma^p_{i-1,2}}$) functions.*

Theorem 3.15 *For $i \geq 1, k \geq 2$ we have $\Delta^c_{i,k} \subseteq \Delta^p_{i,k} \subseteq \Delta^c_{i+1,k}$ and $\Box^c_{i,k} \subseteq \Box^p_{i,k} \subseteq \Box^c_{i+1,k}$.*

PROOF: The inclusions $\Delta^c_{i,k} \subseteq \Delta^p_{i,k}$ and $\Box^c_{i,k} \subseteq \Box^p_{i,k}$ are Lemmas 3.10 and 3.13.

The inclusion $\Delta^p_{i,k} \subseteq \Delta^c_{i+1,k}$ is immediate because $\Delta^p_{i,k} \subseteq \Sigma^p_{i,k}$, for which $\Delta^c_{i+1,k}$ has an oracle.

For the inclusion $\Box^p_{i,k} \subseteq \Box^c_{i+1,k}$, observe that bit extraction is in $\Box^p_{0,2}$, so any desired bit of the $\Box^p_{i,k}$ function can be found in $\Delta^c_{i+1,k}$. It then remains only for $\Box^c_{i+1,k}$ to compute each of these bits in parallel; the uniformity condition can easily specify which bit of the $\Box^p_{i,k}$ function to extract. ∎

4 Bootstrapping R^i_k

Every paper introducing a new theory of bounded arithmetic contains a "bootstrapping" section showing that the theory in question has nice closure properties, can encode sequences efficiently, *etc.* The present paper is no exception. Fortunately, much of the bootstrapping of R^i_k has already been done in [All91].

4.1 DCI, PPIND, LPIND, and LLIND

Theorem 4.1 (Folklore, Allen, Takeuti) *In the presence of the quantifier-free axioms of R^1_2, the following are equivalent:*

- Σ^b_i-*DCI*
- Σ^b_i-*PPIND*
- Σ^b_i-*LPIND*
- Σ^b_i-*LLIND*

PROOF: DCI trivially simulates PPIND, since their conclusions are the same and PPIND has stronger hypotheses.

PPIND can simulate LPIND on the Σ^b_i formula $A(x)$ as follows. Let $B(x) = A(|x| \dot{-} 1)$. Then clearly $A(0) \supset B(0) \wedge B(1) \wedge B(2)$, taking care of the base cases of PPIND on B. For the induction step, observe that

$$B(\mathsf{Fh}(x)) \Rightarrow A(|\mathsf{Fh}(x)| \dot{-} 1) \Rightarrow A\left(\left\lceil \frac{|x|}{2} \right\rceil \dot{-} 1\right) \Rightarrow A\left(\left\lfloor \frac{|x| \dot{-} 1}{2} \right\rfloor\right) \Rightarrow A(|x| \dot{-} 1) \Rightarrow B(x).$$

Thus we can do PPIND on $B(x)$ up to $B(2t+1)$, which implies $A(|t|)$ as desired.

LPIND can simulate LLIND in much the same way, letting $B(x) = A(1 + |x \dot{-} 1|)$ (*i.e.* $B(x)$ is A of the *binary* length of x) and using LPIND on $B(x)$ up to $B(|2t+1|)$.

To simulate DCI with LLIND, we shall interpret a bit string as a balanced binary tree of suitably aligned substrings, and prove by induction on the height of the tree that all vertices of the tree have a certain property. It is a straightforward exercise to define a term $\mathsf{Block}(x, w, i)$ which yields the i-th leftmost vertex of x in the w-th level from the bottom. One can then define, given a Σ^b_i formula $A(x, \vec{y})$, another Σ^b_i formula $B(x, w, \vec{y})$ stating that for all blocks b at level w of the tree interpretation of x, $A(b, \vec{y})$ holds. LLIND on the variable w in $B(x, w, \vec{y})$ then yields the desired result. For further details, see [Blo92], pp. 95–96. ∎

4.2 Adding Symbols by Definition

Definition 4.2 (S. Buss) *The function $f(\vec{x})$ is Σ^b_i-defined in theory T if there is a Σ^b_i formula $A_f(\vec{x}, y)$ (in the language of T) such that $T \vdash (\forall \vec{x})(\exists! y) A_f(\vec{x}, y)$ and $\mathbf{N} \models f(\vec{n}) = m \leftrightarrow A_f(\vec{n}, m)$.*

In particular, if f is Σ_1^b-defined in theory T, then let T^f denote the theory T augmented with the function symbol f (which may appear anywhere, including induction formulæ) and the defining axiom $f(\vec{x}) = y \leftrightarrow A_f(\vec{x}, y)$. S. Buss [Bus86a] pointed out that every Σ_i^b (respectively Π_i^b) formula in T^f is T^f-provably equivalent to a Σ_i^b (resp. Π_i^b) formula not involving f. Thus if all of T's general axiom schemas are applicable to, say, "all Σ_i^b formulæ", then f can be added to the language without changing the applicability of those schemas, and thus T^f is a conservative extension of T.

Definition 4.3 (Buss) *A formula ϕ is Δ_i^b with respect to theory T if there are a Σ_i^b formula ϕ^+ and a Π_i^b formula ϕ^-, both T-provably equivalent to ϕ.*

Buss showed that if R is Δ_1^b-defined in theory T, then every formula of T^R is provably equivalent to a T formula of the same level of syntactic complexity, and so if all of T's general axiom schemas are applicable to "all Σ_i^b formulæ", then T^R is a conservative extension of T.

4.3 Multi-variable induction

The various induction schemes of section 2.4 all apply to only one variable. It is often convenient (especially when Σ_1^b-defining a function of several arguments) to induct on two or more variables at once. [CU93] shows that IS_2^1 can prove the validity of multi-variable PIND, by interleaving the bits of several variables into one. We shall do much the same in R_2^1, although we must change the method of interleaving to work for DCI. For simplicity, we define multi-variable induction only up to terms whose length is a power of two; this will suffice for our purposes, and ensures that repeatedly applying Bh and Fh to said term produces a full, balanced binary tree.

First we Σ_1^b-define a function Ins_1 in R_2^1. This function takes a number and produces a number of exactly twice the dyadic length, formed by inserting a 1 digit after each dyadic digit of the original number. It has the Σ_1^b graph

$$\varphi(x, y) \Leftrightarrow |y| = 2 \cdot |x| \wedge (\forall i < |x|)(\mathsf{Bit}(2 \cdot i, y) = \mathsf{Bit}(i, x) \wedge \mathsf{Bit}(2 \cdot i \dot{-} 1, y) = 1).$$

The existence of such a y can be proven in R_2^1 by a straightforward DCI on x, and so the Ins_1 function may be added conservatively to the language of R_2^1.

We can likewise Σ_1^b-define functions $\mathsf{OddBits}$ and $\mathsf{EvenBits}$, which extract the odd- and even-numbered bits of their one argument respectively; details are left to the reader. We then define (by composition) $\mathsf{RotRight}(x) = \mathsf{Conc}(1, \mathsf{Msp}(x, 1))$ and $\mathsf{Interleave}(x, y) = \mathsf{Ins}_1(x) + \mathsf{RotRight}(\mathsf{Ins}_1(x)) \dot{-} (x\#3)$. One can prove, without induction, that

$$\begin{aligned}
|x| = |y| \ &\supset\ \mathsf{OddBits}(\mathsf{Interleave}(x, y)) = x, \\
|x| = |y| \ &\supset\ \mathsf{EvenBits}(\mathsf{Interleave}(x, y)) = y, \\
|x| = |y| \ &\supset\ \mathsf{Bh}(\mathsf{Interleave}(x, y)) = \mathsf{Interleave}(\mathsf{Bh}(x), \mathsf{Bh}(y)), \text{ and} \\
|x| = |y| \ &\supset\ \mathsf{Fh}(\mathsf{Interleave}(x, y)) = \mathsf{Interleave}(\mathsf{Fh}(x), \mathsf{Fh}(y)).
\end{aligned}$$

If we wish to induct on the variables x_1 and x_2 at once in the Σ_i^b formula $A(x_1, x_2, \vec{y})$, we can do so by defining $B(x, \vec{y})$ as the Σ_i^b formula

$$(\mathsf{PwrTwo}(|x|) \wedge |x| \neq 1) \supset A(\mathsf{OddBits}(x), \mathsf{EvenBits}(x), \vec{y}),$$

and inducting on B instead (up to any value whose length is an even power of two). The precise definition of 2-variable DCI, omitting the free variables $\vec{b}$ which may appear anywhere, is

$$\frac{\Gamma, A(\mathsf{Bh}(a_1), \mathsf{Bh}(a_2)), A(\mathsf{Fh}(a_1), \mathsf{Fh}(a_2)) \longrightarrow A(a_1, a_2), \Delta}{\Gamma, A(0,0), A(\{1,2\} \times \{1,2\}), |t_1| = |t_2|, \mathsf{PwrTwo}(|t_1| \longrightarrow A(t_1, t_2), \Delta}$$

Note that, in order for two-variable DCI to be valid, all five base cases $A(0,0,\vec{b})$, $A(1,1,\vec{b})$, $A(1,2,\vec{b})$, $A(2,1,\vec{b})$, and $A(2,2,\vec{b})$ must hold. The base case $A(0,0,\vec{b})$ implies $B(0,\vec{b})$. The other two base cases of B induction, $B(1,\vec{b})$, and $B(2,\vec{b})$, are vacuous, and the remaining four A base cases are used to prove the length 2 cases of $B(\mathsf{Bh}(x),\vec{b}), B(\mathsf{Fh}(x),\vec{b}) \longrightarrow B(x,\vec{b})$.

Once we can induct on two variables at once, the process can be iterated to handle any fixed number of variables.

4.4 Rounding Lengths

We define a sequence of fast-growing (by bounded arithmetic standards) functions $\mathsf{RoundUp}_j(x)$ inductively as follows:

$$\begin{aligned} \mathsf{RoundUp}_{-1}(x) &= x \\ \mathsf{RoundUp}_{j+1}(x) &= 2(2^{\mathsf{RoundUp}_i(|x|)} - 1) \end{aligned}$$

If $\mathsf{RoundUp}_{j+1}$ is provably total in R^1_k, it is clear (in R^i_k) that $|\mathsf{RoundUp}_{j+1}(x)| = \mathsf{RoundUp}_i(|x|)$, that $\mathsf{RoundUp}_{j+1}(x) \geq x$, and that for $j \geq 0$, all the dyadic digits of $\mathsf{RoundUp}_j(x)$ are 2's (which implies for $j \geq 1$, all the digits of $|\mathsf{RoundUp}_j(x)|$ are 2's; for $j \geq 2$, all the digits of $||\mathsf{RoundUp}_j(x)||$ are 2's; and so on.) Furthermore, $\mathsf{RoundUp}_0(x)$ is the largest value y such that $|y| = |x|$; $\mathsf{RoundUp}_1(x)$ is the largest value y such that $||y|| = ||x||$; and so on.

The functions can be explicitly bounded as follows:

$$\begin{aligned} \mathsf{RoundUp}_{-1}(x) &\leq x \\ |\mathsf{RoundUp}_0(x)| &\leq |x| \\ \mathsf{RoundUp}_0(x) &\leq 2x \\ |\mathsf{RoundUp}_1(x)| &\leq 2|x| \\ \mathsf{RoundUp}_1(x) &\leq (x+1)^2 \\ |\mathsf{RoundUp}_2(x)| &\leq (|x|+1)^2 \\ \mathsf{RoundUp}_2(x) &\leq (2x+1)\#_2(2x+1) \\ |\mathsf{RoundUp}_3(x)| &\leq (2|x|+1)\#_2(2|x|+1) \\ \mathsf{RoundUp}_3(x) &\leq (2(x+1)^2)\#_3(2(x+1)^2) \\ &\vdots \end{aligned}$$

Without writing a precise general formula, it should be clear that $\mathsf{RoundUp}_k$ is bounded by an $\mathcal{L}_k$ term, for all $k \geq 1$. So we may at least hope to Σ^b_1-define $\mathsf{RoundUp}_k$ in IR^1_k.

The $\mathsf{RoundUp}_{-1}$ and $\mathsf{RoundUp}_0$ functions have closed-form definitions, but we also point out, to get us started, that

$$\mathsf{RoundUp}_0(x) = y \Leftrightarrow |y| = |x| \land (\forall i < |y|)(\mathsf{Bit}(i, y) = 2).$$

Then the remaining functions have graphs
$\mathsf{RoundUp}_1(x) = y \;\Leftrightarrow\; ||y|| = ||x|| \wedge$
$(\forall i < ||y||)(\mathsf{Bit}(i, |y|) = 2 \wedge$
$(\forall i < |y|)(\mathsf{Bit}(i, y) = 2);$
$\mathsf{RoundUp}_2(x) = y \;\Leftrightarrow\; |||y||| = |||x||| \wedge$
$(\forall i < |||y|||)(\mathsf{Bit}(i, ||y||) = 2) \wedge$
$(\forall i < ||y||)(\mathsf{Bit}(i, |y|) = 2) \wedge$
$(\forall i < |y|)(\mathsf{Bit}(i, y) = 2),$
and so on. Clearly, all these graphs are Σ^b_0.

For $j \le k$, R^1_k can Σ^b_1-define $\mathsf{RoundUp}_j$ by first Σ^b_1-defining $\mathsf{RoundUp}_{j-1}$ and then proving that $\mathsf{RoundUp}_{j-1}(|x|)$ is bounded by a suitable sharply-bounded term in $\mathcal{L}_k$; then $2^{\mathsf{RoundUp}_{j-1}(|x|)}$ exists and the rest is trivial. Details are left to the reader.

The principal application for these functions in the present paper is to construct suitable values for multi-variable induction. For example, if $y = \mathsf{RoundUp}_1(x)$, then $y \ge x$ and $\mathsf{PwrTwo}(|y|+2)$, so if we let $z = 4y+3$, we can conclude $|z| = |y|+2$ and hence $\mathsf{PwrTwo}(|z|)$.

4.5 Comprehension and Extensionality

Theorem 4.4 $R^1_2 \vdash x \ne y \leftrightarrow (\exists i < \max(|x|, |y|))(\mathsf{Bit}(i, x) \ne \mathsf{Bit}(i, y))$.

This makes it easy to prove the single-valuedness of functions defined bitwise.

PROOF: The $\leftarrow$ direction is trivial by equality axioms. For the $\rightarrow$ direction, we Σ^b_1-define a function $\mathsf{BitDiff}(x, y)$, which returns 0 for $x = y$, and for $x \ne y$ returns the largest i such that $\mathsf{Bit}(i \dot- 1, x) \ne \mathsf{Bit}(i \dot- 1, y)$.

The $\mathsf{BitDiff}$ function has an obvious graph:
$\mathsf{BitDiff}(x, y) = i \;\Leftrightarrow$
$(x = y \wedge i = 0) \vee$
$(\mathsf{Bit}(i \dot- 1, x) \ne \mathsf{Bit}(i \dot- 1, y) \wedge (\forall i < j \le \max(|x|, |y|))(\mathsf{Bit}(j \dot- 1, x) = \mathsf{Bit}(j \dot- 1, y)))).$
The function thus defined is patently single-valued, by trichotomy. To prove that it's total, first observe that if $|x| \ne |y|$, then $i = \max(|x|, |y|)$ satisfies the requirements (since $\mathsf{Bit}(i, \min(x, y)) = 0 \ne \mathsf{Bit}(i, \max(x, y))$).

Otherwise, $|x| = |y|$. We'd like to reason by 2-variable DCI, which only applies to values whose length is a power of 2. Observe that if we let $x' = \mathsf{Conc}(\mathsf{Msp}(4 \cdot \mathsf{RoundUp}_1(x) + 6, |x|), x)$ and $y' = \mathsf{Conc}(\mathsf{Msp}(4 \cdot \mathsf{RoundUp}_1(y) + 6, |y|), y)$, *i.e.* x and y padded on the left with 2's up to length a power of 2, then any correct value for $\mathsf{BitDiff}(x', y')$ is also a correct value for $\mathsf{BitDiff}(x, y)$.

If $x' = y' = 0$, $x' = y' = 1$, or $x' = y' = 2$, then $\mathsf{BitDiff}(x', y') = 0$.

If $x' = 1, y' = 2$ or *vice versa*, then $\mathsf{BitDiff}(x', y') = 1$.

For $|x'| > 1$, assume $\mathsf{BitDiff}(\mathsf{Fh}(x'), \mathsf{Fh}(y')) = i_1$ and $\mathsf{BitDiff}(\mathsf{Bh}(x'), \mathsf{Bh}(y')) = i_2$. If $i_1 \ne 0$, then $i = i_1 + |\mathsf{Bh}(x')|$ works; otherwise $\mathsf{Fh}(x') = \mathsf{Fh}(y')$, so their corresponding bits are equal by equality axioms, so $i = i_2$ works. ∎

For another example of 2-variable DCI, R^1_2 can Σ^b_1-define the OR function, which returns the bitwise disjunction of its two arguments, treating the dyadic digits 1 and 2 as `F` and `T` respectively, and assuming its arguments are of the same, power-of-2, length. OR is proven total by a straightforward DCI in two variables, and single-valued by Theorem 4.4.

We can now Σ_1^b-define a function named $\mathsf{CountDown}$, which takes an argument z such that $\mathsf{PwrTwo}(||z||)$ (a restriction for technical reasons) and returns the concatenation, in reverse lexicographic order, of all $2^{||z||}$ dyadic numbers of length $||z||$. It has the Σ_0^b graph
$\mathsf{CountDown}(z) = w \Leftrightarrow$
$\mathsf{PwrTwo}(||z||) \supset$
$(|w| = 2^{||z||} \cdot ||z|| \wedge (\forall i < 2^{||z||})(\mathsf{Substr}(w, i \cdot ||z||, ||z||) = i + 2^{||z||} - 1))$.
To define it, we first define several auxiliary functions.

- $f_1(z, s) = w \Leftrightarrow$
  $(\neg\mathsf{PwrTwo}(|s|) \wedge w = 0) \vee$
  $(\mathsf{PwrTwo}(|s|) \wedge |w| = 2^{||z||} \cdot |s| \wedge (\forall i < 2^{||z||})(\forall j < |s|)(\mathsf{Bit}(i \cdot |s| + j, w) = 2 \leftrightarrow j = ||z||))$
  can be proven total by DCI on z. Clearly $f_1(0, s) = 0$ and $f_1(1, s) = f_1(2, s) = (s\#_2 1) + 1$. For larger z, let $w' = f_1(\mathsf{Bh}(z), s)$. Then $w = \mathsf{Conc}(\mathsf{Shl}(w', 1), \mathsf{Shl}(w', 1))$ satisfies the definition. Single-valuedness, for $|s|$ a power of 2, is proven by Theorem 4.4; for $|s|$ not a power of 2 it is trivial.

- $f_2(z, s) = w \Leftrightarrow$
  $(\neg\mathsf{PwrTwo}(|s|) \wedge w = 0) \vee$
  $(\mathsf{PwrTwo}(|s|) \wedge |w| = 2^{||z||} \cdot |s| \wedge (\forall i < 2^{||z||})(\mathsf{Substr}(w, i \cdot |s|, |s|) = i + (s\#_2 1)))$
  can likewise be proven total by DCI on z. It is clear that $f_2(0, s) = 0$ and $f_2(1, s) = f_2(2, s) = (s\#_2 1)$. For larger z, let $w_1 = f_1(\mathsf{Bh}(z), s)$ and $w_2 = f_2(\mathsf{Bh}(z), s)$. Then $w = \mathsf{Conc}(\mathsf{OR}(w_1, w_2), w_2)$ satisfies the requirements. Single-valuedness is shown as before.

Now $\mathsf{CountDown}(z)$ can be defined, for $||z||$ a power of 2, simply as $f_2(z, |z|\#_2 1)$. I leave it for the reader to see how this all works.

The following theorems are all straightforward generalizations of theorems in [All91]:

- Given a Δ_i^b predicate $A(x, \vec{y})$, R_k^i Σ_i^b-defines the function $f_A(z, \vec{y})$ whose $|z|$ bits indicate the truth values of $A(0), A(1), \ldots, A(|z| - 1)$.

- R_k^i Σ_1^b-defines efficient functions for encoding and decoding sequences, predicates for testing sequence-hood, and bounds for the size of encoded sequences.

- Successive bounded existential (respectively universal) quantifiers can be collapsed into one, provably in R_k^i.

- R_k^i proves Δ_i^b-replacement, *i.e.* if s and t are any terms not involving x or y, then $(\forall y \leq |s|)(\exists x \leq t)\phi(\vec{a}, x, y)$ implies $(\exists w)(\forall y \leq |s|)(\beta(w, y) \leq t \wedge \phi(\vec{a}, \beta(w, y), y))$, where β is a sequence decoding function.

5 The Buss-Krajíček-Takeuti characterization

In [BKT93] is a characterization of the $\Sigma^b_{i,3}$-definable functions of R^i_3. The computational model in that paper uses "witnessing oracles", which answer an existential query either with "no" or by providing a satisfying value (of quasipolynomial length) for the outermost existential quantifier. Since there may be multiple possible witnesses to a given existential question, the functions defined by invoking these witnessing oracles may be multi-valued. Nevertheless, in certain cases this computational model is equivalent to the parallel model of the present paper.

The main result of this section, Theorem 5.4, is that the function class $\mathcal{FP}_3^{\Sigma^p_{i-1}}[wit, \log^{O(1)}]$ of [BKT93], restricted to single-valued functions, is precisely $\Box^c_{i,3}$. The author originally proved this indirectly, using results of [BKT93] and a flawed version of Theorem 6.1, but then, on a suggestion of S. Buss, discovered the direct (and, incidentally, valid!) proof presented here.

5.1 Background

Definition 5.1 (Buss, Krajíček, Takeuti) *A multi-valued function f is computable in $\mathcal{FP}_3^{\Sigma^p_{i-1}}[wit, \log^{O(1)}]$ iff there is a Turing machine that runs in quasipolynomial time and makes at most polylog many queries to a fixed Σ^p_{i-1} witnessing oracle, such that for all x, M outputs one of the values of $f(x)$.*

The notation $\mathcal{FP}_3^{\Sigma^p_{i-1}}$, without the stuff in brackets, is what I call $\Box^p_{i,3}$. Note that the oracle may (in my notation) be $\Sigma^p_{i-1,k}$ for *any* $k \leq 3$, without changing the class thus defined: the machine making the query can precompute the quantifier bounds and pass them to the oracle as extra parameters.

Definition 5.2 (Buss, Krajíček, Takeuti) *A multi-valued function f is weakly Σ^b_i-defined by theory T iff for some Σ^b_i formula A,*

- $T \vdash (\forall x)(\exists y)A(x,y)$, *and*
- *if $A(n,m)$ holds in the natural numbers, then m is a value of $f(n)$.*

The following theorem was stated in [BKT93] only for $k = 3$ (with "$\log^{O(1)}$" in place of "$||\mathcal{L}_k||$"), but their proof generalizes easily to larger k.

Theorem 5.3 (Buss, Krajíček, Takeuti) *For $i \geq 2$, $k \geq 3$, for any multi-valued function f, the following are equivalent:*

- $f \in \mathcal{FP}_k^{\Sigma^p_{i-1,k}}[wit, ||\mathcal{L}_k||]$
- *f is weakly $\Sigma^b_{i,k}$-defined by theory R^i_k*
- *f is weakly $\Sigma^b_{i,k}$-defined by theory S^{i-1}_k*

(Note that the second requirement in Defn. 5.2 does *not* say "if and only if". Buss, Krajíček, and Takeuti also defined a class called "strong $\mathcal{FP}_3^{\Sigma_{i-1}^p}[wit, \log^{O(1)}]$" and a notion of "strong Σ_i^b-definability" using "if and only if", and proved these equivalent as well, but the difference is only relevant for multi-valued functions and so we shall ignore it henceforth.)

5.2 $\mathcal{FP}_k^{\Sigma_{i-1,k}^p}$ coincides with $\square_{i,k}^c$: the $k \geq 3$ case

Theorem 5.4 *For $i \geq 2$ and $k \geq 3$, the restriction of $\mathcal{FP}_k^{\Sigma_{i-1,k}^p}[wit, ||\mathcal{L}_k||]$ to single-valued functions is $\square_{i,k}^c$.*

Proof: ($\supseteq$): Suppose $F \in \square_{i,k}^c$. Then F is computable by a uniform $\square_{i,k}^c$ circuit family with $\Sigma_{i-1,k}^p$ yes/no oracle ρ.

With at most a constant factor increase in depth, we can assume that the circuit family is levelled, all output gates are at level 0, each even level contains no oracle gates, and each odd level consists entirely of oracle gates. (The last restriction requires excluding constant-valued ρ.) Embed the resulting gates in a rectangular array, indexed by row and column (r, c), in such a manner that a gate's position in the array easily determines its gate number and thence its type. (For example, give every gate at an even level exactly two children, one or both of which it may ignore, and give every gate at an odd level exactly $|t|$ children. Gate (r, c) is descended from output gate $\left\lfloor \frac{c}{(2|t|)^{r/2}} \right\rfloor$ or $\left\lfloor \frac{c}{2(2|t|)^{(r-1)/2}} \right\rfloor$, as r is even or odd respectively, by a path formed by repeatedly modding out 2 and $|t|$ alternately. If $|t|$ is first rounded up to a power of 2, all this is computationally trivial — c's binary form is the concatenation of the output gate number and the path.) Thus the resulting circuit family is still U_{E^*} uniform.

The output of gate (r, c) can be described by a single $\Sigma_{i-1,k}^p$ relation $\sigma(n, r, c, \vec{b})$ where n is the input size and $\vec{b}$ are the outputs of row $r + 1$. (The relation σ simply determines the type of gate (r, c) and simulates it; the most complex case is that of an oracle gate, which is $\Sigma_{i-1,k}^p$.) Define $\phi(n, r, \vec{b}, j)$ to be the formula

$$(\exists |w| = \text{width of row } r)(|w|_1 \geq j) \wedge (\forall c < |w|)(Bit(c, w) \supset \sigma(|x|, r, c, \vec{b}))),$$

where $|w|_1$ represents the number of 1's in w. The ϕ relation is $\Sigma_{i-1,k}^b$ (note $i \geq 2$), and monotone in j, so for any fixed x, we can find the maximum j satisfying ϕ by binary search in $||\mathcal{L}_k||$ time. For this maximum j, the only possible witness w is the string representing *exactly* the values of all the gates at level r (assuming $\vec{b}$ correctly represents the values of all the gates at level $r+1$), because the clause $(\forall c < |w|)(Bit(c, w) \supset \sigma(|x|, r, c, \vec{b}))$ ensures that all the gates whose bit in w is 1 are correctly computed, and if any gate whose bit in w is 0 were incorrectly computed, j wouldn't be maximal.

Thus $||\mathcal{L}_k||$ ϕ-queries, the last one witnessing, suffice to determine the outputs of all gates at level r from the outputs of all gates at level $r + 1$. Iterating this over the $||\mathcal{L}_k||$ levels, starting with the input, allows us to compute the correct values of all the gates at the output level, and thus the value of the function F.

($\subseteq$): Suppose $f \in \mathcal{FP}_k^{\Sigma_{i-1,k}^p}[wit, ||\mathcal{L}_k||]$ and f happens to be single-valued. As shown in [BKT93], we can assume without loss of generality that the quasipolynomial-time machine only uses an ordinary yes/no oracle until its last query, at which point it demands a witness. Furthermore, we can assume without loss of generality that it doesn't demand this witness until it has confirmed by yes/no query that there is one.

Let formula $\psi(x, w)$ assert that there is a computation of the $\mathcal{FP}_k$ machine (up to the time that it demands a witness) in which

- the computation starts with x on the input,
- for all j, if the j-th most significant bit of w is 1, then the j-th oracle query in the computation answers "yes",
- the behavior of the machine is consistent with the alleged tape contents and alleged oracle answers in the computation, and
- all the "yes" oracle answers (at most $||\mathcal{L}_k||$) in the computation are correct.

The formula $\psi(x, w)$ is $\Sigma^b_{i-1,k}$ (note $i \geq 2$), consisting of a bounded existential quantifier around some sharply-bounded universal quantifiers around a $\Sigma^p_{i-1,k}$ oracle. Furthermore, ψ is monotone in each bit of w. Since w has only $||\mathcal{L}_k||$ many bits, we can find the lexicographically maximum w satisfying ψ by deterministic binary search. Having found this w, for each output bit r in parallel we query the $\Sigma^p_{i-1,k}$ oracle $\psi'(x, w, r)$, which asserts that there is a computation in which

- the computation starts with x on the input,
- for all j, the j-th most significant bit of w is exactly the answer to the j-th oracle query in the computation,
- the behavior of the machine is consistent with its alleged tape contents and the alleged oracle answers,
- all the "yes" oracle answers in the computation are correct, and
- there exists a witness to the final query that leads the machine to output a value with bit r set.

By maximality of w, there is a unique and correct such computation until the final witnessing query. Since we assumed f was single-valued, all the output bits will be consistent, even if the circuits to compute different bits happen to find different witnesses to the final oracle query.

Technically, a $\square^c_{i,k}$ circuit family contains only one oracle, and we have used oracle gates for both ψ and ψ'. But as usual, both can be replaced by oracles for your favorite $\Sigma^p_{i-1,k}$-complete problem (under $\square^c_{1,k}$ reductions). Thus $f \in \square^c_{i,k}$. ∎

5.3 The $k = 2$ case

[Kra93] shows that the predicates in $\mathcal{P}^{\Sigma_{i-1}^p}[O(\log)]$ (*i.e.*, $\mathcal{P}$ with $O(\log)$ queries to a Σ_{i-1}^p oracle) are those Σ_i^b-definable in S_2^{i-1}. His technique actually uses $\mathcal{FP}^{\Sigma_{i-1}^p}[wit, O(\log)]$, but the restriction to predicates rather than functions allows him to eliminate the witnessing and multivaluedness at the last moment. Thus in the notation of [BKT93], he essentially proved

Theorem 5.5 (Krajíček) *For $i \geq 2$, for any multi-valued function f, the following are equivalent:*

- $f \in \mathcal{FP}^{\Sigma_{i-1,2}^p}[wit, O(\log)]$
- *f is weakly $\Sigma_{i,2}^b$-defined by theory S_2^{i-1}*

Interestingly enough, the proof of Theorem 5.4 can be adapted for $k = 2$:

Theorem 5.6 *For $i \geq 2$, the restriction of $\mathcal{FP}^{\Sigma_{i-1}^p}[wit, O(\log)]$ to single-valued functions is $\Box_{i,2}^c$.*

Proof: The proof of Theorem 5.4 relies on $k \geq 3$ in several ways, but these can all be fixed.

($\supseteq$): The previous proof takes $||\mathcal{L}_k||$ time to do a binary search for each of the $||\mathcal{L}_k||$ rows of the array. Since $||\mathcal{L}_2||$ is not closed under multiplication, we can no longer afford this. However, by our convention, for some term $t \in \mathcal{L}_2$, each oracle gate has "depth" $||t|| = \theta(\log(n))$ (recall that t is nontrivial). But a $\Box_{i,2}^c$ circuit has depth $O(\log(n))$, so there must be a constant bound d on the number of oracle gates along any one path. Arrange the array in d meta-layers, each containing one row of oracle gates and $O(\log)$ rows of ordinary gates. We define a $\Delta_{0,2}^c$ relation $\sigma'(n, r, c, \vec{b})$ that does the same thing as σ for rows containing only ordinary gates. We then define a $\Sigma_{1,2}^p$ relation $\phi'(n, r, \vec{b})$ by $(\exists |w| = \text{width of row } r)(\forall c < |w|)(Bit(c, w) \leftrightarrow \sigma'(|x|, r, c, \vec{b})))$. The algorithm is then as before, but for ordinary rows we simply make a witnessing query to ϕ' rather than doing a binary search on ϕ.

($\subseteq$): We can still assume that an $\mathcal{FP}^{\Sigma_{i-1}^p}[wit, O(\log)]$ machine demands witnesses only on its final query, by a lemma in [Kra93]. The rest of the proof is unaltered. ∎

Note that Theorems 5.4 and 5.6 are unlikely to be improved to $i = 1$: in the latter case, such a result would imply $P = ALOGTIME$.

5.4 Applications and extensions

Corollary 5.7 *For $i \geq 2$, $k \geq 2$,*

$$\Delta_{i,k}^c = \mathcal{P}_k^{\Sigma_{i-1}^p}[||\mathcal{L}_k||].$$

In other words, $DTIME(|\mathcal{L}_k|)$ with $||\mathcal{L}_k||$ queries to a Σ_{i-1}^p oracle is equal to $ATIME(||\mathcal{L}_k||)$ with unlimited queries to the same oracle.

This provides a new proof of, and generalizes, a result of [BH91].

Corollary 5.8 *For $i \geq 2$, $k \geq 2$, the single-valued functions weakly $\Sigma^b_{i,k}$-definable in S^{i-1}_k are precisely $\square^c_{i,k}$.*

We can in fact improve on Corollary 5.8 by eliminating the word "weakly". The following direct proof shows that the $\square^c_{i,k}$ functions can actually be $\Sigma^b_{i,k}$-defined in the usual, single-valued sense in S^{i-1}_k.

Lemma 5.9 *For $i \geq 2$, $k \geq 2$, S^{i-1}_k can $\Sigma^b_{i,k}$-define all the $\square^c_{i,k}$ functions.*

PROOF: Let $f \in \square^c_{i,k}$. Then f is computed by some U_{E^*}-uniform circuit family of genus k circuits with a fixed $\Sigma^p_{i-1,k}$ oracle X. The existence of a computation to construct and simulate this circuit, with correct oracle queries, is certainly $\Sigma^b_{i,k}$, but we must show that S^{i-1}_k can *prove* both existence and uniqueness of the result. I'll describe the algorithm first, then formalize it in S^{i-1}_k. First, assume the circuits are treelike (which is OK for $\square^c_{i,k}$) and levelled. We'll scan from left to right across the leaf layer, at each step computing the values of all the gates in the whole circuit whose descendant leaves are among the ones we've visited. For each leaf, this requires adding on at most the unique path from that leaf to the root, which is length $||\mathcal{L}_k||$, and we can find the lexicographically maximum (and hence correct) string of this length.

More formally, let $\mathsf{Anc}(r, x)$ be the number of gates in the circuit for x, all of whose descendant leaves are numbered r or lower. Define $\mathsf{InfoTuple}(w, r, x)$ to be the Σ^b_{i-1} formula asserting that

- w is an r-tuple of bit strings of lengths $\mathsf{Anc}(1, x)$, $\mathsf{Anc}(2, x) - \mathsf{Anc}(1, x)$,…,$\mathsf{Anc}(r, x) - \mathsf{Anc}(r-1, x)$ respectively, and
- for all $j < |w_r|$, if the j-th most significant bit of w_r is T, then the j-th ancestor gate of leaf r outputs T when given input as specified in w.

Thus $\mathsf{InfoTuple}$ asserts that w is a possible assignment of values to the ancestors of leaves $1, \ldots, r$, in which at least all the "yes" answers are correct for their presumed inputs.

Define $\mathsf{GtrTuple}(w, w')$ to be a Δ^b_0 formula asserting that for some r, r',

- w and w' are tuples of r and r' elements respectively,
- $r \geq r'$,
- $(\forall j < r')(w_j \geq w'_j)$, and
- $(r > r') \vee (\exists j < r')(w_j > w'_j)$.

That is, each bit-string entry of w, treated as a dyadic number, is at least as great as the corresponding entry of w', and either at least one entry of w is *greater* than the corresponding entry of w' or w is longer than w'.

Now define $\mathsf{InfoSeq}(W, r, x)$ to be the Σ^b_{i-1} formula[3] asserting that

[3]this may look Π^b_i, but in practice $r \leq |W|$ so the universal quantifiers can be made sharply bounded.

- W is a sequence of exactly r tuples,
- $(\forall j < r)\mathsf{InfoTuple}(W_j, j, x)$, and
- $(\forall j < r-1)\mathsf{GtrTuple}(W_{j+1}, W_j)$.

By LIND on $(\exists W, r)\mathsf{InfoSeq}(W, r, x)$, S^i_k can prove there is a maximum-length W satisfying $\mathsf{InfoSeq}(W, r, x)$ with r bounded by the number of leaves in the circuit; note that each tuple has $|\mathcal{L}_k|$ entries, each entry is bounded in numeric value by $|\mathcal{L}_k|$, and each tuple in such a sequence must have a distinct set of entries. Let w denote the last element of this maximum-length W. It must contain a bit-string for each leaf, or w (and therefore W) could be extended by adding on a string of zeroes for the next leaf. The first bit-string, w_1, must have the maximum possible value consistent with input x, or W could be extended by increasing w_1. Thus all the "no" answers in w_1 must be correct. Similarly, w_2 must have the maximum possible value consistent with x and w_1, and so on; after $|\mathcal{L}_k|$ steps we conclude that every gate value in w is correct. Extracting the value of the function is then straightforward.

Since every step in the above algorithm is clearly single-valued, and the loop implicit in the above paragraph has at most $|\mathcal{L}_k|$ iterations, S^{i-1}_k proves the uniqueness of this value. ∎

Corollary 5.10 *For $i \geq 2$, $k \geq 2$, the single-valued functions $\Sigma^b_{i,k}$-definable in S^{i-1}_k are precisely $\Box^c_{i,k}$.*

Corollary 5.11 *For $i \geq 2$, $k \geq 2$, if T^{i-1}_k is conservative over S^{i-1}_k with respect to $\forall\exists\Sigma^b_{i,k}$ sentences (*i.e. *if $S^{i-1}_k \equiv T^{i-1}_k$), then $\Box^c_{i,k} = \Box^p_{i,k}$.*

6 Links to Bounded Arithmetic

In [Blo95], I stated that $\Box^c_{i,k}$ comprised exactly the functions $\Sigma^b_{i,k}$-definable in R^i_k. I am no longer convinced that $\Box^c_{i,k}$ realizes all R^i_k proofs. However, in this section we prove that $\Box^c_{i,k}$ is at least *contained* in the set of functions thus definable.

Theorem 6.1 *For $i \geq 1, k \geq 3$, any single-valued function $f \in \Box^c_{i,k}$ is $\Sigma^b_{i,k}$-definable in R^i_k.*
For $i \geq 1, k = 2$, any single-valued function $f \in \mathcal{FNC}^{\Sigma^p_{i-1,2}}$ is $\Sigma^b_{i,2}$-definable in R^i_2.

This is proven by methods similar to those in [Bus86a, All91]. First, $\Box^c_{i,k}$ (respectively $\mathcal{FNC}^{\Sigma^p_{i-1,2}}$) is equal to a certain recursively-defined function algebra. Second, the base functions of this algebra are Σ^b_i-definable in R^i_k, and the operators preserve such Σ^b_i-definability, so any function in the algebra is Σ^b_i-definable in R^i_k.

6.1 A function algebra

We define a hierarchy of function algebras $\{A_{i,k}\}_{i\geq 0, k\geq 2}$ by applying two simple operations to a fixed collection of base functions on the universe $\{\texttt{F}, \texttt{T}\}^*$, which we identify with the natural numbers as described in section 2.1.

Definition 6.2 *The* BASE *functions are the following:*

- *the constant functions* $\texttt{F}$, $\texttt{T}$, *and* λ, *which return the single-bit strings* $\texttt{F}$ *and* $\texttt{T}$ *and the empty string (*i.e. *the dyadic numbers 1, 2, and 0), respectively;*
- $\mathsf{LSP}(x, y)$, *the rightmost $|y|$ bits of x, or simply x if $|y| > |x|$.*
- $\mathsf{MSP}(x, y)$, *all but the rightmost $|y|$ bits of x, or the empty string (*i.e. 0*) if $|y| > |x|$.*
- $\mathsf{BIT}(i, x)$, *the i-th rightmost bit of x (treating i as a dyadic number),*
- $\mathsf{COND}(w, x, y, z) = \begin{cases} x & \text{if } w = \lambda \\ y & \text{if } \mathsf{LSP}(w, 1) = \texttt{F} \\ z & \text{if } \mathsf{LSP}(w, 1) = \texttt{T} \end{cases}$
- $\mathsf{CONC}(x, y)$, *the concatenation of x and y,*
- $\mathsf{BH}(x)$, *the rightmost $\left\lfloor \frac{|x|}{2} \right\rfloor$ bits of x,*
- $\mathsf{FH}(x)$, *the leftmost $\left\lceil \frac{|x|}{2} \right\rceil$ bits of x,*
- $\mathsf{NOT}(x)$, *the bitwise complement* [4] *of x,*
- $\mathsf{OR}(x, y)$, *the bitwise disjunction of x and y (undefined if $|x| \neq |y|$),*

[4] which can also be written numerically as $3 \cdot (2^{|x|} \dot{-} 1) \dot{-} x$. Obviously, the bitwise definition is clearer. We shall not even attempt to give numerical definitions of OR, AND, $\mathsf{INS}_{\texttt{F}}$, or $\mathsf{INS}_{\texttt{T}}$!

- $\mathsf{AND}(x, y)$, *the bitwise conjunction of* x *and* y *(undefined if* $|x| \neq |y|$*),*
- $\mathsf{INS}_{\mathrm{F}}(x)$, *a string exactly twice as long as* x *such that for all* $i < |x|$, *we have* $\mathsf{BIT}(2i, \mathsf{INS}_{\mathrm{F}}(x)) = \mathrm{F}$ *and* $\mathsf{BIT}(2i+1, \mathsf{INS}_{\mathrm{F}}(x)) = \mathsf{BIT}(i, x)$,
- $\mathsf{INS}_{\mathrm{T}}(x)$, *defined analogously with* $\mathsf{BIT}(2i, \mathsf{INS}_{\mathrm{T}}(x)) = \mathrm{T}$.

Note that the functions MSP and LSP behave somewhat differently from their namesakes in Section 2.2. MSP and LSP consider only the *length* of their second argument, rather than treating the argument itself as a dyadic number. As a result, all the BASE functions have length determined by the lengths of their arguments (which makes it easier to envision multi-output circuit families computing them).

Definition 6.3 *A function* f *is defined by* $|\mathcal{L}_k|$-bounded divide-and-conquer recursion *(or* DCR*) from functions* g *and* h *if there is a term* $t \in \mathcal{L}_k$ *such that*

$$f(z, b, \vec{x}) = \begin{cases} g(z, \vec{x}) & \text{if } |z| \leq |b|, \\ h(z, b, \vec{x}, f(\mathsf{FH}(z), b, \vec{x}), f(\mathsf{BH}(z), b, \vec{x})) & \text{otherwise,} \end{cases}$$

and $|f(z, b, \vec{x})| \leq |t(z, b, \vec{x})|$ *for all* $z, b, \vec{x}$.

Note that this form of recursion does *not* necessarily go all the way down to 0; instead, the extra parameter b tells it when to bottom out. This extra parameter appears necessary in order to do many useful things with DCR; see [Blo94].

We extend this notion of divide-and-conquer recursion in the obvious way to define constantly many functions by *simultaneous* recursion:

Definition 6.4 *Functions* $f_1, \ldots, f_j$ *are defined by* simultaneous $|\mathcal{L}_k|$-bounded divide-and-conquer recursion *from functions* $g_1, \ldots, g_j$ *and* $h_1, \ldots, h_j$ *if there is a term* $t \in \mathcal{L}_k$ *such that for all* $1 \leq i \leq j$,

$$f_i(z, b, \vec{x}) = \begin{cases} g_i(z, \vec{x}) & \text{if } |z| \leq |b|, \\ h_i(z, b, \vec{x}, \vec{f}(\mathsf{FH}(z), b, \vec{x}), \vec{f}(\mathsf{BH}(z), b, \vec{x})) & \text{otherwise,} \end{cases}$$

and $|f_i(z, b, \vec{x})| \leq |t(z, b, \vec{x})|$ *for all* $z, b, \vec{x}$.

Don't confuse this with the multi-variable induction defined in section 4.3. First, this is a recursion scheme for defining functions, rather than an induction scheme for proving theorems. Second, this recursion scheme still applies to only one variable at once; it is called "simultaneous" because it defines several *functions* in parallel, in terms of one another.

Definition 6.5 *The function algebra* $A_{i,k}$ *is the closure of* BASE *and the 1-bit characteristic functions of* $\Sigma^c_{i-1,k}$ *relations*[5] *under the operations of composition and simultaneous* $|\mathcal{L}_k|$-*bounded DCR.*

[5]where $\Sigma^p_{-1,k}$ is by convention $\{\}$

For example, we define a function $\mathsf{COUNTUP}(z) \in A_{0,2}$ which produces the concatenation of all $2^{||z||}$ bit-strings of length $||z||$, in lexicographic order. (This example not only shows some of DCR's power, but proves useful later on.)

Let $\mathsf{SHL}(x, y) = \mathsf{LSP}(\mathsf{CONC}(x, y), x)$; this appends y to x, discarding high bits so the result still has length $|x|$.

Let $\mathsf{FALSES}(x) = \mathsf{AND}(x, \mathsf{NOT}(x))$, a string of $|x|$ F's.

Let

$$\begin{aligned} \mathsf{LENGTH}'(x) &= \begin{cases} \lambda & \text{if } |x| = 0 \\ \mathsf{CONC}(\mathsf{LENGTH}'(\mathsf{BH}(x))), 1) & \text{otherwise;} \end{cases} \\ \mathsf{LENGTH}(x) &= \mathsf{MSP}(\mathsf{LENGTH}'(\mathsf{CONC}(x, 1)), 1) \end{aligned}$$

$\mathsf{LENGTH}(x)$ is a string of 1's of the same length as the dyadic representation of $|x|$; equivalently, $\mathsf{LENGTH}(x)$ is $||x||$ written in unary.

Let

$$\mathsf{MASK}(z, b) = \begin{cases} \lambda & \text{if } |z| = 0 \\ \mathsf{SHL}(\mathsf{FALSES}(b), 2) & \text{if } |z| = 1 \\ \mathsf{CONC}(\mathsf{SHL}(\mathsf{MASK}(\mathsf{BH}(z), b), 1), \mathsf{SHL}(\mathsf{MASK}(\mathsf{BH}(z), b), 1)) & \text{if } |z| > 1, \end{cases}$$

$$\mathsf{AUX}(z, b) = \begin{cases} \lambda & \text{if } |z| = 0 \\ \mathsf{FALSES}(b) & \text{if } |z| = 1 \\ \mathsf{CONC}(\mathsf{AUX}(\mathsf{BH}(z), b), \mathsf{OR}(\mathsf{MASK}(\mathsf{BH}(z), b), \mathsf{AUX}(\mathsf{BH}(z), b))) & \text{if } |z| > 1, \end{cases}$$

and finally $\mathsf{COUNTUP}(z) = \mathsf{AUX}(z, \mathsf{LENGTH}(z))$. I leave seeing how this works as an exercise for the reader.

Lemma 6.6 *For $i \geq 0$ and $k \geq 3$, $\square^c_{i,k} = A_{i,k}$.*
For $i \geq 0$ and $k = 2$, $\square^c_{i,k} \subseteq A_{i,k} \subseteq \mathcal{FNC}^{\Sigma^p_{i-1,k}}$.

PROOF: ($A_{i,k} \subseteq \square^c_{i,k}$ or $FNC^{\Sigma^p_{i-1,k}}$): For $i = 0$, all the BASE functions are clearly in $\square^c_{0,k}$. By Lemma 3.5, $\square^c_{0,k}$ is closed under composition, and it is not hard to see that it's closed under $|\mathcal{L}_k|$-bounded DCR too. (This requires $k \geq 3$ so circuit depth is closed under multiplication. For the $k = 2$ case, since $\mathcal{FNC}$ is closed under polynomial-bounded DCR [All91], we can conclude $A_{0,2} \subseteq \mathcal{FNC}$.)

For $i > 0$, recall that the characteristic functions of $\Sigma^p_{i-1,k}$ relations are in $\square^c_{i,k}$ (and $\mathcal{FNC}^{\Sigma^p_{i-1,k}}$) by definition. The proof that $A_{i,k} \subseteq \square^c_{i,k}$ (or for $k = 2$, $A_{i,2} \subseteq \mathcal{FNC}^{\Sigma^p_{i-1,2}}$) is almost exactly as before.

($\square^c_{i,k} \subseteq A_{i,k}$): First, note that it suffices to get $\Delta^c_{i,k} \subseteq A_{i,k}$. Suppose $f \in \square^c_{i,k}$; then its bit-graph is in $\Delta^c_{i,k}$ (and hence, as we shall show, in $A_{i,k}$). To get the whole function value at once, we use $\mathsf{COUNTUP}(f(\vec{x}))$ to build a list, in lexicographic order, of all the $||f(\vec{x})||$-bit strings. Interpreting these in binary would give a list of the numbers $0, 1, \ldots, 2^{||f(\vec{x})||}-1$, which we could use as inputs to the bit-graph to get the individual bits, then concatenate them all together with a DCR on CONC. The technique is spelled out in more detail in [Blo94].

Unfortunately, we've defined BIT and bit-graphs to interpret the bit position as *dyadic*, and $\mathsf{COUNTUP}(f(\vec{x}))$ interpreted as a list of dyadic numbers gives $2^{||f(\vec{x})||}-1, \ldots, 2^{||f(\vec{x})||+1}-2$. We get around this by using not the bit-graph of f itself, but rather the bit-graph of $\hat{f}(\vec{x}) = f(\vec{x}))\ \mathrm{F}^{2^{||f(\vec{x})||}-1}$, *i.e.*, f with $2^{||f||}-1$ F's appended at the end; this is no problem for the U_{E^*} uniformity criterion, so $\hat{f}$ is still $\Box^c_{i,k}$ and the above reasoning goes through.

($\Delta^c_{i,k} \subseteq A_{i,k}$): For $i = 0$, suppose $f(\vec{x}) \in \Delta^c_{0,k}$ (which is simply $ATIME(||\mathcal{L}_k||)$). At the cost of a constant factor in time, we can assume the ATM is *strictly* alternating between existential and universal states and that it accesses its input tape only at leaves. In [Blo94], the author showed how to simulate the computation of such an ATM within a function algebra similar to $A_{0,k}$. In brief: $A_{0,k}$ can construct a binary tree data structure called PATHS, each of whose leaves contains an encoding of the path from the root to that leaf. This binary tree represents the computation tree of the ATM, so we define a function $\mathsf{EVALTREE}$ by constant-bounded DCR on the tree, bottoming out upon reaching a single leaf, to combine the accept/reject decisions of the various ATM configurations into one. When this recursion bottoms out, we apply another function $\mathsf{EVALLEAF}$, defined by $|\mathcal{L}_k|$-bounded simultaneous DCR to compute the state and the contents of all the work tapes in parallel, to the encoded path to determine the ATM configuration at the corresponding leaf of the computation tree and decide whether it accepts. Thus $\Delta^c_{0,k} \subseteq A_{0,k}$.

The $i > 0$ case is a little more complex, and the $k = 2$ case of it must be treated separately. For $k \geq 3$, recall from Lemmas 2.19 and 3.8 that every $\Box^c_{i,k}$ function's bit-graph is in $ATIME(||\mathcal{L}_k||)$ with oracle states for $\Sigma^p_{i-1,k}$ relations. At the cost of a $||t||$ factor in time, we assume the levels of the ATM rotate strictly among universal, existential, exclusive-or, and oracle states. Let t denote the bounding term for all the oracle states in the machine, rounded up so that $|t|$ is a power of 2. We construct a PATHS tree of depth equal to the time bound of the ATM (counting oracles as taking $||t||$ time, as usual), and apply a modified version of $\mathsf{EVALTREE}$, which at its base (*i.e.* a single leaf) invokes the $\mathsf{EVALLEAF}$ function exactly as for $i = 0$.

As defined in [Blo94], $\mathsf{EVALTREE}$ returned a bit indicating whether a certain subtree of the ATM's computation tree accepted or rejected. In this setting, it will ultimately do the same, but it may have intermediate values up to $|t(x)|$ bits long. The following algorithm describes the recursive phase of $\mathsf{EVALTREE}$:

let v_1 and v_2 be $\mathsf{EVALTREE}(\mathsf{FH}(T), b, x)$ and $\mathsf{EVALTREE}(\mathsf{BH}(T), b, x)$ respectively.
let r be the remainder of $||T||$ mod $(||t|| + 3)$.
If $0 \leq r < ||t|| - 1$, return the concatenation of strings v_1 and v_2.
If $r = ||t|| - 1$, v_1 and v_2 should be strings of length $|t|/2$ each; concatenate them and return ρ of the resulting string, where ρ is the oracle for this circuit.
If $r = ||t||$, v_1 and v_2 should be single bits; return their AND.
If $r = ||t|| + 1$, v_1 and v_2 should be single bits; return their OR.
If $r = ||t|| + 2$, v_1 and v_2 should be single bits; return their XOR.

Since $|T|$ is cut in half at each iteration, $||T||$ indicates the depth of a tree, up to an additive constant. Of every $||t|| + 3$ levels of the PATHS tree, $||t||$ are used to accumulate the answers from the $|t|$ children of a given oracle configuration, one computes XOR, one OR, and one AND. The final value of EVALTREE is 1 or 0, indicating whether the ATM accepts or rejects. The value of a multi-bit function can be pieced together just as in the $i = 0$ case.

The algorithm described above can be implemented in $A_{i,k}$: use divide and conquer recursion to construct the unary numbers $||T||$ and $||t||$, then divide $||T||$ by $||t|| + 3$ in unary by repeated subtraction (using the MSP function), which takes at most $||T||$ iterations and therefore can be done by DCR on T. With the remainder in unary, we can then distinguish the five cases with a few MSP and COND functions.

For $k = 2$, $||\mathcal{L}_k||$ isn't closed under multiplication, so we can't afford to make the ATM rotate strictly among layers of universal, existential, exclusive-or, and oracle states. Instead, as in Theorem 5.6, we bound the number of oracle queries along any computation path by a constant d and arrange the ATM (at the cost of a constant factor of time) into d meta-layers each containing one layer of oracle states and $||\mathcal{L}_k||$ layers of other states, which (at the cost of another constant factor of time) we force to rotate strictly among universal, existential, and exclusive-or layers. The EVALTREE function changes accordingly: we give it an additional argument M, in which we give it the precomputed unary number $||PATHS||/d$, *i.e.* the depth of each metalayer, and at each step of the divide and conquer recursion we do the following:

let v_1 and v_2 be EVALTREE(FH(T)$, b, x, M$) and EVALTREE(BH(T)$, b, x, M$) respectively.
let r_1 be the remainder of $||T||$ mod M
If $0 \leq r_1 < ||t|| - 1$, return the concatenation of strings v_1 and v_2.
If $r_1 = ||t|| - 1$, v_1 and v_2 should be strings of length $|t|/2$ each; concatenate them and return ρ of the resulting string, where ρ is the oracle for this circuit.
Otherwise, let r_2 be the remainder of $r_1 - ||t||$ mod 3.
If $r_2 = 0$, v_1 and v_2 should be single bits; return their AND.
If $r_2 = 1$, v_1 and v_2 should be single bits; return their OR.
If $r_2 = 2$, v_1 and v_2 should be single bits; return their XOR.

Again, the numbers $||T||$, M, r_1, and r_2 are all available in unary, so we can do arithmetic on them easily. ∎

This concludes the proof of lemma 6.6.

Lemma 6.7 *For $i \geq 1$, $A_{i,2} = \mathcal{FNC}^{\Sigma^p_{i-1},2}$.*

PROOF: Lemma 6.6 has already given us the $\subseteq$ direction. For the $\supseteq$ direction, which resembles the proof of Theorem 1.3.3 of [All91], let $f \in \mathcal{FNC}^{\Sigma^p_{i-1},2}$ be computed by a (wolog, levelled) circuit family of depth $O(\log^j(n))$. We reason by induction on j.

If $j = 1$, then $f \in \Box^c_{i,2}$, and therefore $f \in A_{i,2}$ by Lemma 6.6. If $j > 1$, think of an f circuit as divided into $O(\log(n))$ meta-layers of depth at most $\log^{j-1}(n)$ each. Let g be a function which, given the values of all the inputs to a meta-layer and a number indicating which meta-layer it is, computes all the outputs from that meta-layer. This function can be computed in depth $O(\log^{j-1}(n))$ with a Σ^p_{i-1} oracle, even allowing for constructing a copy of the relevant part of the f circuit, so by the inductive hypothesis $g \in A_{i,2}$. The desired function f can then be computed by iterating g $O(\log(n))$ times, which we can do by DCR. Since the f circuit is only polynomial size, each value of f and g need only be polynomially many bits long, so the DCR is polynomially bounded. ∎

6.2 Definability in R_k^i

The main result of this section is the following:

Theorem 6.8 *For $i \geq 1$, $k \geq 2$, if $f \in A_{i,k}$, then f is $\Sigma_{i,k}^b$-definable in R_k^i.*

Proof: If f is the characteristic function of a $\Sigma_{i-1,k}^c$ relation, then it is clearly $\Sigma_{i,k}^b$-definable, without induction. If f is one of the functions F, T, λ, MSP, LSP, BIT, COND, CONC, BH, or FH, then its encoding is $\Sigma_{0,2}^b$-definable, without induction, in the language of R_k^i. The same holds of NOT, less obviously: the definition is $\lambda x.(3 \cdot (x\#1) \dot{-} x)$.

The $\mathsf{INS}_{\mathrm{F}}$ function has the Σ_0^b graph

$$\varphi(x, y) \Leftrightarrow |y| = |x| + |x| \wedge (\forall i < |x|)(\mathsf{Bit}(i + i, y) = 1 \wedge \mathsf{Bit}(i + i + 1, y) = \mathsf{Bit}(i, x)).$$

It can be proven total by a straightforward DCI on the Σ_1^b formula $(\exists y \leq (x + 1)^3)\varphi(x, y)$, and proven single-valued by Theorem 4.4.

The definition of $\mathsf{INS}_{\mathrm{T}}$ is similar. The AND and OR functions likewise have obvious Σ_0^b graphs based on their bitwise definitions; they can be proven total by 2-variable DCI and single-valued by Theorem 4.4.

Thus all the BASE functions, and the characteristic functions of $\Sigma_{i-1,k}^b$ relations, are $\Sigma_{i,k}^b$-definable in R_k^i. The class of functions $\Sigma_{i,k}^b$-definable in R_k^i is trivially closed under composition, so we need only to prove it closed under $|\mathcal{L}_k|$-bounded DCR.

Lemma 6.9 *For $i \geq 1$, $k \geq 2$, if g and h are $\Sigma_{i,k}^b$-definable in R_k^i, and f is defined by $|\mathcal{L}_k|$-bounded DCR on g and h, then f is $\Sigma_{i,k}^b$-definable in R_k^i.*

Proof: Suppose R_k^i Σ_i^b-defines g and h by the formulæ $\phi_g(z, \vec{x}, y)$ and $\phi_h(z, b, \vec{x}, u_1, u_2, y)$ respectively, and f is defined from g and h with a length bound of $|s|$. We define a formula $\phi_f(z, b, \vec{x}, y)$ that $\Sigma_{i,k}^b$-defines f in R_k^i by asserting the existence of an encoded binary tree T representing the computation. At vertex i of this binary tree we'll store data of the form $\langle y_i, z_i \rangle$ (of length $O(|s| + |z|)$) to assert that $F(z_i, b, \vec{x}) = y_i$. The root block must be equal to $\langle y, z \rangle$, each block $\langle y_i, z_i \rangle$ with $|z_i| > |b|$ must be appropriately related to its children by ϕ_h, and each block $\langle y_i, z_i \rangle$ with $|z| \leq |b|$ must satisfy $\phi_g(z_i, \vec{x}, y_i)$.

In order to ensure that R_k^i can prove the existence of such a T, we represent T in a somewhat inefficient, "inflated" way: to represent a tree of depth $||z||$ with $O(|s| + |z|)$ bits of information at each vertex, we use an encoded bit-string of $4^{||z||-1}$ blocks, each of length $O(|s| + |z|)$. Each subtree is represented by a string whose first half contains the data for the subtree's root, padded with garbage bits of unspecified value. The third quarter of the string represents the left subtree, and the fourth quarter the right subtree. This enables us to apply divide-and-conquer induction easily to the whole tree; for details of the techniques, see [Blo94]. Note that even in inflated form, the length of T is a polynomial in $|z|$ and $|s|$, so this construction works for $k \geq 2$. ∎

Theorem 6.8 follows immediately. ∎

6.3 Conservation, pro and con

[BKT93] showed the following:

Theorem 6.10 (Buss,Krajíček,Takeuti) *For $i \geq 2$, $k \geq 3$, R^i_k is $\forall\exists\Sigma^b_{i,k}$-conservative over S^{i-1}_k.*

Lemma 6.7 and Theorem 6.8, together with Corollary 5.8, gives us some insight as to why it has been so difficult to strengthen Theorem 6.10 to $k = 2$:

Theorem 6.11 *For $i \geq 2$, if R^i_2 is conservative over S^{i-1}_2 with respect to $\forall\exists\Sigma^b_{i,2}$ sentences, then $\mathcal{FNC}^{\Sigma^p_{i-1,2}} = \square^c_{i,2}$.*

PROOF: Every $FNC^{\Sigma^p_{i-1,2}}$ function is R^i_2-definable. If it were also S^{i-1}_2-definable, it would be in $\square^c_{i,2}$. ∎

Recall that in Corollary 5.7, we showed a collapse of $\mathcal{P}$ with $O(\log)$ queries (or, equivalently, logspace — see [BH91]) down to NC^1, relative to oracles in each level of the polynomial hierarchy. Recall also that Wilson [Wil87] showed $\mathcal{NC} = \mathcal{NC}^1$ in the presence of a suitable *recursive* oracle. By the above theorem, both of these results could be strengthened to "$\mathcal{NC} = \mathcal{NC}^1$ relative to any Σ^p_{i-1}-hard oracle" if $S^{i-1}_2 \prec_{\Sigma^b_{i,2}} R^i_2$.

7 Links to Complexity Hierarchies

The authors of [KPS90, KPT91] defined a notion of interactive computation called a "student-teacher game", a dialogue between an omniscient teacher and a computationally limited student assigned to find the optimal solution to some problem. Each time the student presents a suboptimal solution, the teacher replies with "no, that's not optimal; see, here's a better solution", and the student may then use this better solution in constructing the next solution. (In fact, the student may simply parrot it back *as* the next solution. This algorithm is called the *trivial student.*) With the aid of theorem 6.1 and this computational model, we can prove several results analogous to those of [KPT91].

7.1 A KPT witnessing theorem for S^i_k

Theorem 7.1 *For $i \geq 1, k \geq 3$, and $\phi \in \exists\Pi^b_{i,k}$, if $S^i_k \vdash (\exists y)(\forall z)\phi(a,y,z)$, then there are $\Box^c_{i+1,k}$ functions $f_1, \ldots f_r$ such that*

$$S^i_k \vdash \phi(a, f_1(a), b_1) \vee \phi(a, f_2(a, b_1), b_2) \vee \ldots \vee \phi(a, f_r(a, b_1, \ldots b_{r-1}), b_r).$$

That is, there is a $\Box^c_{i+1,k}$ student that finds a witness for $(\exists y)(\forall z)\phi(a,y,z)$ within some constant number r of rounds, provably in S^i_k.

Proof: The proof resembles that of Theorem A of [KPT91], with two main differences. First, the equational theory contains a function symbol for every definition in $A_{i+1,k}$ rather than in a Cobham-style algebra for $\mathcal{FP}^{\Sigma^p_i}$. Second, prefix induction is replaced by divide-and-conquer induction, and successor induction by prefix induction, throughout.

Define an equational theory $CV_{i+1,k}$ with a function symbol for each function definition in $A_{i+1,k}$. Among the functions thus defined is, for every open formula $\psi(x) \in \Sigma^b_{i,k}$, the function h_ψ defined by $h_\psi(s) = \mathsf{LEFT}(h'_\psi(\mathsf{COUNTUP}(s), s, |s|), s)$, where

$$h'_\psi(T,s,b) = \begin{cases} 0 & \text{if } |T| \leq |b| \wedge \neg\psi(\mathsf{LEFT}(T,s)) \\ T+1 & \text{if } |T| \leq |b| \wedge \psi(\mathsf{LEFT}(T,s)) \\ \max(h'_\psi(\mathsf{Fh}(T),s,b), h'_\psi(\mathsf{Bh}(T),s,b)) & \text{otherwise} \end{cases}$$

and $\mathsf{LEFT}(n,s)$ denotes the leftmost n bits of s, or all of s if $n > |s|$. The intent is that if $\psi(\lambda)$ holds, then $h'_\psi(\mathsf{COUNTUP}(s), s, |s|)$ returns one more than the maximal length of a prefix p of s such that $\psi(p)$ holds. We shall normally invoke this function with ψ monotone, $\psi(\lambda)$ true, and $\psi(s)$ false, so $h_\psi(s)$ returns the minimal prefix p of s such that $\psi(s)$ does *not* hold. Note, to satisfy the definitions, that $|h_\psi(T,s,b)|$ is bounded by $|T+1|$, that $\mathsf{LEFT} \in \Box^c_{0,2} = A_{0,2}$, and that $\psi \in \Sigma^b_{i,k} \subseteq \Delta^c_{i+1,k}$, and so ψ's characteristic function is in $\Box^c_{i+1,k} = A_{i+1,k}$.

The theory $CV_{i+1,k}$ is axiomatized with some reasonable set of open axioms about BASE (see, *e.g*, [Blo92, Blo97]) as well as the following "induction" axiom for every h_ψ of the above form:

$$\psi(0) \wedge \neg\psi(s) \wedge h_\psi(s) = c \supset \psi\left(\left\lfloor \frac{c}{2} \right\rfloor\right) \wedge \neg\psi(c).$$

Note that this axiom implies ψ-PIND, and so $CV_{i+1,k}$ is an extension of S^i_k. Furthermore, it is a *conservative* extension, as shown by the following two lemmas:

Lemma 7.2 *For $i \geq 1, k \geq 3$, $\psi \in \Sigma^b_{i,k}$, S^i_k $\Sigma^b_{i+1,k}$-defines h_ψ.*

PROOF: The function graph $h'_\psi(T, s, b) = c$ can be defined by

$(\exists w)(w$ is a binary tree of depth $||s||$, each node containing an ordered pair with lengths bounded by $|T|$ and $|s|$ respectively, and
the root of the tree is $\langle T, c\rangle$, and
for each leaf $\langle u, v\rangle$,

$$u = \begin{cases} \lambda & \text{if } \neg\psi(\mathsf{LEFT}(v,s) \\ \mathsf{LEFT}(v+1,s) & \text{if } \psi(\mathsf{LEFT}(v,s), \text{ and} \end{cases}$$

for each non-leaf node $\langle u, v\rangle$ with children $\langle u_l, v_l\rangle$ and $\langle u_r, v_r\rangle$,
$v = \mathsf{CONC}(v_l, v_r)$ and
$u = \max(u_l, u_r)$.

The whole formula is $\Sigma^b_{i+1,k}$. R^{i+1}_k can prove the totality of the function by DCI on T, so by Theorem 6.10 S^i_k can also prove totality. The uniqueness of h_ψ is easy by proving the uniqueness of w. ∎

Lemma 7.3 *For $i \geq 1, k \geq 2$, and $\psi \in \Sigma^b_{i,k}$, the theory S^i_k proves the "induction" axiom for h_ψ.*

PROOF: Consider the $\Sigma^b_{i+1,k} \cap \Pi^b_{i+1,k}$ formula

if w is as in the above definition of the graph of $h'_\psi(T, s, b)$,
and if $\psi(0)$ but $\neg\psi(s)$, then
for all nodes $\langle u, v\rangle$ in w,
$u = 0 \Leftrightarrow \neg\psi(\mathsf{LEFT}(\mathsf{LEFT}(|b|, v), s)$,
$u = \mathsf{RIGHT}(|b|, v) + 1 \Leftrightarrow \psi(\mathsf{LEFT}(\mathsf{RIGHT}(|b|, v), s)$, and
$1 \leq u \leq \mathsf{RIGHT}(|b|, v) \Leftrightarrow \psi\left(\lfloor \frac{u}{2} \rfloor\right) \wedge \neg\psi(u)$.

By DCI on T in the above formula, we can prove it for $T = \mathsf{COUNTUP}(s)$. The root of any such w will be $\langle T, c\rangle$ for some c. The assumption $\psi(0)$ rules out the possibility $c = 0$, and the assumption $\neg\psi(s)$ rules out the possibility $c = \mathsf{RIGHT}(|b|, T) + 1$ (since $\mathsf{RIGHT}(|b|, T) \geq |s|$), so only the third case remains. By [Bus87], S^i_k proves LIND, *a fortiori* DCI, on $\Sigma^b_{i+1} \cap \Pi^b_{i+1}$ formulæ. ∎

Thus $CV_{i+1,k}$ is a conservative extension of S^i_k. Now suppose $S^i_k \vdash (\exists x)(\forall y)\phi(a, x, y)$ where $\phi \in \exists\Pi^b_{i,k}$. Then $CV_{i+1,k}$ proves it too, and by the same Herbrand argument as in [KPT91], $CV_{i+1,k}$ proves

$$\phi(a, f_1(a), b_1) \vee \phi(a, f_2(a, b_1), b_2) \vee \ldots \vee \phi(a, f_r(a, b_1, \ldots b_{r-1}), b_r)$$

for some constantly many $A_{i+1,k}$ functions $f_1, \ldots, f_r$. By conservativity of $CV_{i+1,k}$ over S^i_k, S^i_k defines the same functions $f_1, \ldots, f_r$, which are all $\Box^c_{i+1,k}$, and proves the same statement about them. This concludes the proof of Theorem 7.1. ∎

7.2 Collapsing Bounded Arithmetic

A previous version of this paper contained an adaptation of the argument of [KPT91] that $T_2^i = S_2^{i+1} \Rightarrow \Sigma_{i+1}^p \subseteq \Delta_{i+1}^p/\text{poly}$. However, Buss [Bus94] gave a simpler proof of a stronger result, so I'll adapt that technique instead. The main substantive difference between the following proof and Buss's is the replacement of prefix induction with divide and conquer induction; however, the proof technique is interesting enough to warrant another exposition.

Theorem 7.4 *For $i \geq 1$, $k \geq 3$, if $S_k^i = R_k^{i+1}$, then*

- *S_k^i proves that every Σ_{i+1}^b formula $B(x)$ is equivalent to a formula of the form $C(x, A(x))$ where A is a $(\Sigma_{i+2}^b \cap \Pi_{i+2}^b)$-defined function of S_k^i depending only on $|x|$, and $C \in \Pi_{i+1}^b$;*
- *S_k^i proves $\Sigma_{i+1,k}^p \subseteq \Delta_{i+1,k}^p/poly$;*
- *$S_k^i = S_k^{i+1}$;*
- *$S_k^i = S_k$, which is therefore finitely axiomatized; and*
- *for every bounded formula in $\mathcal{L}_k$, S_k^i proves it equivalent to a Boolean combination of $\Sigma_{i+2,k}^b$ formulæ.*

The idea, as in other proofs [KPT91, Bus94] involving student-teacher games, is to play the omniscient teacher, giving away as little information as possible, and playing for long enough that the student *must* show some semblance of originality. Later, a *non*-omniscient teacher faced with a new problem can enlist the aid of this same demonstrably nontrivial student and solve the problem. But before we can apply this technique, we must define some preliminary notions.

Definition 7.5 (Buss) *Π_i^B is the class of quantified Boolean formulæ in prenex form containing i blocks of like quantifiers, starting with a $\forall$.*

$TRU^i(\phi, w)$ is a formula stating that ϕ encodes a Π_i^B formula and w encodes a satisfying assignment of its free variables.

$SAT^i(\phi)$ is the formula $(\exists w \leq \phi) TRU^i(\phi, w)$.

Note that TRU^i and SAT^i are formulæ in the language of bounded arithmetic, while ϕ is a numeric *encoding* of a quantified Boolean formula (but we shall often identify it with that quantified Boolean formula, for convenience). Note also that we assume the encodings of ϕ and w are reasonably efficient, encodable and decodable in ALOGTIME, and that any assignment of the free variables in ϕ has an encoding numerically less than ϕ itself. We assume standard functions for encoding and decoding sequences, *e.g.* $\mathsf{SeqLen}(s) =$ the length of the sequence encoded by s.

It is well known that SAT^i is complete for $\Sigma_{i+1,2}^p$, and therefore for $\Sigma_{i+1,2}^b$ formulæ, under polynomial-time, many-one reductions. The techniques of the proof are not computationally demanding, and can be formalized in T_2^i, S_2^i, or even (with care) R_2^i, and the reductions can be weakened to $\mathcal{NC}$ or even ALOGTIME. Indeed, for $k \geq 2$, SAT^i is complete for $\Sigma_{i+1,k}^p$

and $\Sigma^b_{i+1,k}$ under $\square^c_{0,k}$ many-one reductions; the larger growth rates are needed only during the reduction. And by standard techniques, TRU^i is Δ^b_{i+1} with respect to R^i_2.

The problem we shall set our student is, given a sequence of n encoded formulæ $\phi_1, \phi_2, \ldots, \phi_n$, to find the longest initial sequence $\phi_1, \phi_2, \ldots, \phi_m$ all of which are satisfiable, and produce witnesses $w_1, w_2, \ldots, w_m$ for them. (To avoid funny behavior at 0, we give the student a witness w_1 for ϕ_1 for free.) However, we'll accept a witnessed initial sequence if it is *within a factor of 2* in length of the longest one. That is, the student is to realize the principle:

$$(*) \qquad (\forall \phi_1, \ldots, \phi_n \leq 2^{|a|})(\exists w_1, \ldots, w_m \leq \vec{\phi})\mathsf{MaxSoln}(a, \vec{\phi}, \vec{w})$$

where $\mathsf{MaxSoln}(a, \vec{\phi}, \vec{w})$ is defined by

$$\begin{aligned}
&\mathsf{SeqLen}(\vec{w}) \leq \mathsf{SeqLen}(\vec{\phi}) \land \\
&\quad (\forall j \leq \mathsf{SeqLen}(\vec{\phi}))\phi_j \leq 2^{|a|} \land \\
&\quad (\forall j \leq \mathsf{SeqLen}(\vec{w}))w_j \leq \phi_j \land \\
&\quad (\forall j \leq \mathsf{SeqLen}(w))TRU^i(\phi_j, w_j) \land \\
&\quad (\mathsf{SeqLen}(\vec{\phi}) \geq 2 \cdot \mathsf{SeqLen}(w) \supset \exists j \leq 2 \cdot \mathsf{SeqLen}(w))\neg SAT^i(\phi_j).
\end{aligned}$$

Note that $\mathsf{MaxSoln} \in \Pi^b_{i+1}$, so (*) is a $\forall\exists\Pi^b_{i+1}$ statement, and provable in R^{i+1}_k by maximizing $p \leq ||\vec{\phi}||$ in the Σ^b_{i+1} formula

$$(\exists \vec{w} \leq \vec{\phi})(|\mathsf{SeqLen}(\vec{w})| = p \land (\forall j \leq \mathsf{SeqLen}(\vec{w}))TRU^i(\phi_j, w_j)).$$

If, as we shall assume for the rest of this theorem, $S^i_k = R^{i+1}_k$ and therefore S^i_k could also prove principle (*), then Theorem 7.1 would imply the existence of a $\square^c_{i,k}$ student who witnesses it in some constant number r of rounds. If in a given round, the student proposes a sequence of witnesses $w_1, \ldots, w_m$ which is *not* "nearly maximal", the teacher is obliged to demonstrate that by providing witnesses for at least the first $2m$ formulæ. Thus $S^i_k = R^{i+1}_k$ implies there are functions $f_1, \ldots, f_r$, Σ^b_{i+1}-defined in S^i_k, such that, for any witnesses $\vec{w}$ for $\vec{\phi}$,

$$\begin{aligned}
&\mathsf{MaxSoln}(a, \vec{\phi}, f_1(a, \vec{\phi}, w_1)) \lor \\
&\mathsf{MaxSoln}(a, \vec{\phi}, f_2(a, \vec{\phi}, w_1, \ldots, w_{2m_1})) \lor \\
&\mathsf{MaxSoln}(a, \vec{\phi}, f_3(a, \vec{\phi}, w_1, \ldots, w_{2m_2})) \lor \\
&\quad \vdots \\
&\lor \mathsf{MaxSoln}(a, \vec{\phi}, f_r(a, \vec{\phi}, w_1, \ldots, w_{2m_{r-1}}))
\end{aligned}$$

where $m_j = \mathsf{SeqLen}(f_j(a, \vec{\phi}, w_1, \ldots, w_{2m_{j-1}}))$.

Define an *original witness* to be any witness the student produces without first being given. The trivial student, the one who never produces an original witness, will yield $m_1 = 1$, $m_2 = 2$, $m_3 = 4, \ldots, m_r = 2^{r-1}$. So to coax an original witness from our student, it suffices to let $n \geq 2^r$ and make all the formulæ satisfiable. Define the formula $\mathsf{FindsOrigBy}(a, m, \langle \phi_1, \ldots, \phi_s \rangle)$ to be

$$
\begin{aligned}
&(\forall \vec{w} \leq \vec{\phi} \leq 2^{|a|})((\forall j \leq \mathsf{SeqLen}(\vec{\phi}))TRU^i(\phi_j, w_j)) \supset \\
&\quad (\mathsf{SeqLen}(f_1(a, \vec{\phi}, \langle w_1 \rangle)) > 1 \vee \\
&\quad\ \mathsf{SeqLen}(f_2(a, \vec{\phi}, \langle w_1, w_2 \rangle)) > 2 \vee \\
&\quad\ \mathsf{SeqLen}(f_3(a, \vec{\phi}, \langle w_1, w_2, w_3, w_4 \rangle)) > 4 \ \vee \ldots \\
&\quad \vee \mathsf{SeqLen}(f_m(a, \vec{\phi}, \langle w_1, \ldots, w_{2^{m-1}} \rangle)) > 2^{m-1}))
\end{aligned}
$$

This states that, no matter what witnesses are provided for the formulæ $\vec{\phi}$, the game will produce at least one original witness within the first m rounds, and therefore an original witness for one of the first 2^m formulæ. The assumption $S^i_k = R^{i+1}_k$ implies

$$
S^i_k \vdash (\forall \phi_1, \ldots, \phi_{2^r} \leq 2^{|a|}) \left((\bigwedge_{j=1}^{2^r} SAT^i(\phi_j)) \supset \mathsf{FindsOrigBy}(a, r, \vec{\phi}) \right).
$$

It is conceivable (assuming $\mathsf{FindsOrigBy}(a, m, \vec{\phi})$) that, although the game is guaranteed to give an original witness for one of the first 2^m formulæ, the student uses the later formulæ to find it; for example, suppose the last formula in the sequence encodes a polynomial-time algorithm for SAT! The truth value of $\mathsf{FindsOrigBy}(a, m, \vec{\phi})$ may therefore depend on all of $\vec{\phi}$, even though only the first 2^m ϕ's are actually treated as formulæ to be witnessed. We therefore distinguish between the parts of $\vec{\phi}$ that might be witnessed and the parts serving only as "advice" to the computation, and write, *e.g*, $\mathsf{FindsOrigBy}(a, m, \langle \phi_1, \ldots, \phi_{2^m} \rangle, A)$.

Now consider the formula $\mathsf{PreAdvice}(a, m, A)$ defined by

$$
(\forall \phi_1, \ldots, \phi_{2^m} \leq 2^{|a|}) \left((\bigwedge_{j=1}^{2^m} SAT^i(\phi_j)) \supset \mathsf{FindsOrigBy}(a, m, \vec{\phi}, A) \right).
$$

This states that, for any sequence of 2^m satisfiable formulæ $\vec{\phi}$, and any witnesses for those formulæ, the game with advice A will produce an original witness for one of $\vec{\phi}$.

Recall that

$$
S^i_k \vdash (\forall \phi_1, \ldots, \phi_{2^r} \leq 2^{|a|}) \left((\bigwedge_{j=1}^{2^r} SAT^i(\phi_j)) \supset \mathsf{FindsOrigBy}(a, r, \vec{\phi}) \right).
$$

Rephrasing this in terms of advice, $S^i_k \vdash \mathsf{PreAdvice}(a, r, \langle\rangle)$. Since r is a constant, S^i_k proves without induction that there is a minimum $m \leq r$ such that $(\exists A)\mathsf{PreAdvice}(a, m, A)$. Let $\mathsf{Advice}(a, m, A)$ assert that m is such a minimum, and A is a corresponding advice string. Then $S^i_k \vdash (\exists m, A)\mathsf{Advice}(a, m, A)$. Finally, given a formula $\psi \in \Sigma^B_i$ and "auxiliary" formulæ $\phi_1, \ldots, \phi_{2^{m-1}}$, let $\mathsf{TestSeq}(\vec{\phi}, \psi)$ be the sequence of length 2^m formed by concatenating $\vec{\phi}$ with 2^{m-1} copies of ψ.

Now we are ready to prove the theorem. I claim that S^i_k proves

$$
\begin{aligned}
&(\mathsf{Advice}(a, m, A) \wedge \psi \leq 2^{|a|}) \supset \\
&\ (SAT^i(\psi) \leftrightarrow \\
&\quad (\forall \phi_1, \ldots, \phi_{2^{m-1}} \leq 2^{|a|})(\textstyle\bigwedge_{j=1}^{2^{m-1}} SAT^i(\phi_j) \supset \mathsf{FindsOrigBy}(a, m, \mathsf{TestSeq}(\vec{\phi}, \psi), A).
\end{aligned}
$$

PROOF: Assume $\mathsf{Advice}(a,m,A)$ and $\psi \le 2^{|a|}$. If ψ is satisfiable, then for any satisfiable sequence $\phi_1,\ldots,\phi_{2^{m-1}}$, $\mathsf{TestSeq}(\vec{\phi},\psi)$ is a sequence of 2^m satisfiable formulæ and so, by $\mathsf{PreAdvice}(a,m,A)$, an original witness will be found for it.

On the other hand, if ψ is not satisfiable, then any original witness for $\mathsf{TestSeq}(\vec{\phi},\psi)$ must actually witness one of the first 2^{m-1} formulæ $\vec{\phi}$, using ψ only as extra advice. Thus $\mathsf{FindsOrigBy}(a,m,\mathsf{TestSeq}(\vec{\phi},\psi),A)$ is equivalent to $\mathsf{FindsOrigBy}(a,m-1,\vec{\phi},\langle A,\psi\rangle)$. However, by minimality of m, there is *no* advice A' for which $\mathsf{PreAdvice}(a,m-1,A')$ holds. In particular, for the advice $A'=\langle A,\psi\rangle$, there must exist a satisfiable sequence $\phi_1,\ldots,\phi_{2^{m-1}}$ such that $\mathsf{FindsOrigBy}(a,m-1,\vec{\phi},A')$ is false. ∎

Thus in the presence of the "advice" A, which depends only on a, the Σ^b_{i+1}-complete formula $\mathsf{SAT}^i(\psi)$ is equivalent to the Π^b_{i+1} formula

$$(\forall \phi_1,\ldots,\phi_{2^{m-1}} \le 2^{|a|})\left(\bigwedge_{j=1}^{2^{m-1}} SAT^i(\phi_j) \supset \mathsf{FindsOrigBy}(a,m,\mathsf{TestSeq}(\vec{\phi},\psi),A)\right).$$

This proves the first part of the theorem: if $B(x)\in\Sigma^b_{i+1}$, then by the completeness of SAT^i, there is a poly-time function f such that $B(x)\leftrightarrow SAT^i(f(x))$. We can think of the advice A as a (not necessarily single-valued) function of $a\approx 2^{|f(x)|}$, $\Sigma^p_{i+2}\cap\Pi^p_{i+2}$-defined in S^i_k, so we have $B(x)\leftrightarrow C(f(x),A(f(x)))$ where $C(\vec{\phi},A)$ is the above Π^b_{i+1} formula.

The second part follows immediately. In fact, since the reduction function f above can be made quasilinear [Sch78], we get $\Sigma^p_{i+1,k}\subseteq\Delta^p_{i+1,k}/\text{quasilinear}$.

The third, fourth, and fifth parts follow much as in [Bus94].

8 Conclusions

We have completed analogies with several previous results on the theories S_2^i and T_2^i, filling the following gaps in our understanding of bounded arithmetic:

- a general collapse of $\mathcal{P}[\log]$ down to $\mathcal{NC}^1$, relative to all nontrivial levels of the polynomial hierarchy, and analogues for quasipolynomial, *etc.* hierarchies (see Corollary 5.7)
- new complexity-theoretic implications of $S_k^i \equiv T_k^i$ (see Corollary 5.11)
- an elegant computational characterization of the single-valued Σ_i^b-consequences of S_k^{i-1} (see Corollary 5.8)
- complexity-theoretic implications of extending the known $\forall\exists\Sigma_i^b$-conservativity between S_k^{i-1} and R_k^i down to $k = 2$ (see Theorem 6.11)
- complexity-theoretic implications of $S_k^i \equiv R_k^{i+1}$ (see Theorem 7.4)

This paper arose from the desire for more elegant computational characterizations of the $\Sigma_{i,k}^b$ consequences of R_k^i and S_k^{i-1}. The classes $\square_{i,k}^c$ are natural, and seem to comprise a "large" subset of those consequences, but only under the explicit assumption of single-valuedness. Let $\Phi_{i,k}$ denote an (as yet undescribed) computational class that captures *exactly* those consequences, without restricting to single-valued functions. Several research directions present themselves:

- Augment the computational class $\square_{i,k}^c$ with a symmetry-breaking primitive which, given a list of numbers at least one of which has a desired property, chooses an arbitrary element with that property. The resulting class would certainly include $\Phi_{i,k}$, but would the reverse inclusion hold, *i.e.* would the primitive preserve R_k^i definability? Such a symmetry-breaking primitive, if not carefully restricted, would allow arbitrary bounded minimization and hence expand beyond $\square_{i,k}^c$ to all of $\square_{i,k}^p$. Assuming this can be avoided, is there another, more elegant way to describe the class?
- Find complete problems for the levels of the polynomial hierarchy that are "parallel self-reducible", *i.e.* an $\mathcal{FNC}$ or $\square_{1,k}^c$ circuit family with a yes/no oracle for the problem can construct a witness for its outermost existential quantifier. In this case, $\Phi_{i,k}$ would simply be $FNC^{\Sigma_{i-1,k}^p}$ or $\square_{i,k}^c$ respectively.

Either approach would imply Conjecture 1.1. Either approach would allow us to describe $\Phi_{i,k}$ computationally and show, directly from its computational definition, that it can Σ_i^b-realize R_k^i proofs, and thus that it captures all the Σ_i^b consequences of R_k^i. We could then continue the analogy from known results on S_2^i and T_2^i to say

- $\forall\exists\Sigma_i^b$-conservativity between S_k^i and R_k^i would imply $\Phi_{i,k} = \square_{i,k}^p$.
- $\forall\exists\Sigma_i^b$-conservativity (*i.e.* equivalence) between S_k^{i-1} and T_k^{i-1} would likewise imply $\Phi_{i,k} = \square_{i,k}^p$.

On the other hand, perhaps Conjecture 1.1 is simply not true, and $\Phi_{i,k}$ is fundamentally and irremediably a class of multi-valued functions. If we still have an interest in single-valued functions, then, this would suggest that R_k^i is "too big", and that perhaps it should be reformulated, *e.g.* without existential instantiation (or its sequent-calculus analogue).

All the results presented in this paper can also be proven for a suitably axiomatized *intuitionistic* theory IR_k^i — indeed, the axiomatization in Section 2.5 was chosen to be intuitionistically useful. If one of the above approaches were successful, it seems likely that, by techniques similar to [Bus86b, CU93, Har92], $\Phi_{i,k}$ would capture *all* the consequences of IR_k^i, regardless of their syntactic complexity.

Several other holes remain in the web of analogies. One is our inability so far to prove analogues of Theorems 7.1 and 7.4 for $k = 2$, precisely the case of the greatest computational interest. Another is whether R_2^{i+1} is indeed $\forall\exists\Sigma_{i+1}^b$-conservative over S_2^i.

[Kra93] shows that the equivalence $S_2^i \equiv T_2^i$, like the equivalence $S_2^{i+1} \equiv T_2^i$, would have surprising complexity-theoretic implications: any problem solvable in $\mathcal{P}^{\Sigma_{i-1}^p}$ could be solved with only $O(\log)$ many queries to its oracle. (It is not known whether this would imply the collapse of the polynomial hierarchy.) The author has not yet found a natural complexity class corresponding to the Σ_{i+1}^b-definable functions of R_k^i. Such a class would provide stronger (and presumably still less plausible) implications of $R_k^i \equiv S_k^i$.

The larger questions not addressed in this paper remain stubbornly unsolved: *are* there any equivalences among the theories R_k^i, S_k^i, and T_k^i, *is* the theory $T_k = S_k = R_k$ finitely axiomatizable for any k, and *do* the corresponding complexity classes collapse?

Acknowledgments

I would like to thank Sam Buss for introducing me to the subject, for suggesting possible strengthenings of my preliminary results, for pointing me to related work in the literature, and (somewhat ruefully) for pointing out the error in the previous version of the paper. Thanks also to Judy Goldsmith, who provided postdoc support in the early stages of the project.

References

[All91] William Allen. Arithmetizing uniform NC. *Annals of Pure and Applied Logic*, 53(1):1–50, 1991. See also *Divide and Conquer as a Foundation of Arithmetic*, Ph.D. thesis, University of Hawaii at Manoa, 1988.

[BH91] Samuel R. Buss and Louise Hay. On truth-table reducibility to SAT. Information and Computation, 91:86–102, 1991.

[BKT93] Samuel R. Buss, Jan Krajíček, and Gaisi Takeuti. Provably total functions in bounded arithmetic theories R^i_3, U^i_2 and V^i_2. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 116–161. Oxford University Press, 1993.

[Blo92] Stephen Bloch. *Divide and Conquer in Parallel Complexity and Proof Theory.* PhD thesis, University of California, San Diego, 1992.

[Blo94] Stephen Bloch. Function-algebraic characterizations of log and polylog parallel time. *computational complexity*, 4(2):175–205, 1994. See also *Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, 193–206, 1992.

[Blo95] Stephen Bloch. On parallel hierarchies and R^i_k. In Daniel Leivant, editor, *Proceedings of Workshop on Logic and Computational Complexity*, number 960 in Lecture Notes in Computer Science. Springer-Verlag, 1995.

[Blo97] Stephen Bloch. Parameter-free induction in bounded arithmetic. In preparation for submission, 1997.

[Bus86a] Samuel R. Buss. *Bounded Arithmetic.* Number 3 in Studies in Proof Theory. Bibliopolis (Naples), 1986.

[Bus86b] Samuel R. Buss. The polynomial hierarchy and intuitionistic bounded arithmetic. In *Structures Proceedings*, number 223 in Lecture Notes in Computer Science, pages 77–103. Springer-Verlag, 1986.

[Bus87] Samuel R. Buss. Axiomatizations and conservation results for theories of bounded arithmetic. In *Proceedings of a Workshop in Logic and Computation*, AMS Contemporary Mathematics, May 1987.

[Bus94] Samuel R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. Manuscript, 1994.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[Clo89] Peter Clote. A first order theory for the parallel complexity class NC. Technical Report BCCS-8901, Boston College, 1989.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science II*, pages 24–30. North-Holland, 1965.

[CT92] Peter Clote and Gaisi Takeuti. Bounded arithmetic for NC, $ALogTIME$, L and NL. *Annals of Pure and Applied Logic*, 56:73–117, 1992.

[CU93] Stephen A. Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63(2):103–200, September 1993.

[Fag74] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard M. Karp, editor, *Complexity of Computations*, volume 7 of *SIAM-AMS Proceedings*, pages 43–73. 1974.

[Har92] Victor Harnik. Provably total functions of intuitionistic bounded arithmetic. *Journal of Symbolic Logic*, 57(2):466–477, June 1992.

[Hoo83] Jay Hook. *A many-sorted approach to predicative mathematics*. PhD thesis, Princeton University, 1983.

[KPS90] Jan Krajíček, Pavel Pudlák, and Jiři Sgall. Interactive computations of optimal solutions. In *Mathematical Foundations of Computer Science*, volume 452 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 1990.

[KPT91] J. Krajíček, P. Pudlák, and G. Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.

[Kra93] Jan Krajíček. Fragments of bounded arithmetic and bounded query classes. *Transactions of the AMS*, 338(2):587–598, August 1993.

[Nel86] Edward Nelson. *Predicative Arithmetic*. Princeton University Press, 1986.

[Par71] Rohit J. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36:494–508, 1971.

[Ruz81] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[Sch78] C.P. Schnorr. Satisfiability is quasilinear complete in NQL. *Journal of the ACM*, 25:136–145, 1978.

[Wil85] Christopher B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.

[Wil87] Christopher B. Wilson. Relativized NC. *Math. Systems Theory*, 20:13–29, 1987.