# Strengths and weaknesses of LH arithmetic

Chris Pollett
Department of Mathematics,
University of California, Los Angeles, CA cpollett@math.ucla.edu

Randall Pruim
Department of Mathematics and Statistics
Calvin College, Grand Rapids, MI rpruim@calvin.edu

*February 20, 2003– Draft*

**Abstract**

In this paper we provide a new arithmetic characterization of the levels of the log-time hierarchy (LH). We define arithmetic classes $\Sigma_k^{\log}$ and $\Pi_k^{\log}$ that correspond to $\Sigma_k$-LOGTIME and $\Pi_k$-LOGTIME respectively. We break $\Sigma_k^{\log}$ and $\Pi_k^{\log}$ into natural hierarchies of subclasses $\Sigma_k^{m\cdot\log}$ and $\Pi_k^{m\cdot\log}$. We then define bounded arithmetic deduction systems $(T\Sigma_k^{\log})'$ whose $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-definable functions are precisely $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$. We show these theories are quite strong in that (1) *LIOpen* proves for any fixed $m$ that $\Sigma_k^{m\cdot\log} \neq \Pi_k^{m\cdot\log}$, (2) $TAC^0$, a theory that is slightly stronger than $\cup_k(T\Sigma_k^{\log})'$ whose $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-definable functions are LH, proves LH is not equal to $\Sigma_m$-TIME$(s)$ for any $m > 0$ where $2^s \in L$, $s(n) \in \omega(\log n)$, and (3) $TAC^0$ proves $\mathsf{LH} \neq \mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma^k_{\log}$ for all $k$ and $m$. We then show that the theory $TAC^0$ cannot prove the collapse of the polynomial hierarchy. Thus any such proof, if it exists, must be argued in a stronger systems than ours.

*Mathematics Subject Classification:* 03F30, 68Q15
*Keywords:* bounded arithmetic, independence, feasible lower bounds

## 1 Introduction

One way to quantify the difficulty of P = NP problem would be to exhibit a logical theory that is capable of formalizing current attempts at answering this question but is not powerful enough to prove or disprove this equality.

Razborov [15] has argued that most current circuit lower bound techniques can be formalized in certain bounded arithmetic theories. Nevertheless, exhibiting any bounded arithmetic theory which one can demonstrate cannot prove the collapse of the polynomial hierarchy is nontrivial. The first nonconditional result in this direction was given in Pollett [13]. Unfortunately, the theory $Z$ given there was in the original language of bounded arithmetic which has the symbol for multiplication. Care had to be taken to make sure the theory was too weak to manipulate constant-depth, polynomial sized circuits in this language since, if it could, it could reason about $\mathsf{TC}^0$ and almost nothing is known about lower bounds for this class. This resulted in a theory that was so weak it seemed unlikely it could formalize any interesting circuit lower bounds. Despite this, some limited attempt to show one can translate proofs from stronger theories into meaningful results in $Z$ was given in Pollett [14]. In this paper, we use a weaker language for bounded arithmetic than that given in Buss [3]. This allows us to use stronger induction and comprehension principles in defining theories and yet still get a theory that cannot prove the collapse of the polynomial hierarchy.

One of the most celebrated lower bound results known is the result that constant depth, unbounded fan-in circuits of AND's, OR's, and NOT's (i.e., the class $\mathsf{AC}^0$) cannot define parity. Thus, $\mathsf{AC}^0$ does not contain all $p$-time functions. A commonly used uniform version of this class is the log-time hierarchy denoted $\mathsf{LH}$. In this paper, we chose our weaker language so that we could give a new arithmetic characterization of $\mathsf{LH}$. Unlike the characterization of $\mathsf{LH}$ using $\mathsf{FO}$, the levels of our characterization match up with $\mathsf{LH}$ even at the $\mathsf{NLOGTIME}$ and $\mathsf{co\text{-}NLOGTIME}$ levels. To see that this might be difficult and involve a careful choice of initial language notice it is not even known (as far as the authors can determine) whether equals is in $\mathsf{NLOGTIME}$. Using our characterization we then define proof systems $(T\Sigma_k^{\log})'$ whose bounded $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-definable functions are precisely those functions computable in $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$. Here $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$ means we allow an outer length bounded universal quantifier on something which is a Boolean combination of $\Sigma_k^{\log}$-formulas. We show none of these systems is strong enough to prove the polynomial hierarchy collapses by exhibiting a theory $ZAC^0$, which contains $TAC^0 = \cup_k T\Sigma_k^{\log} \supseteq \cup_k (T\Sigma_k^{\log})'$, which cannot prove the collapse of the polynomial hierarchy. The theory $TAC^0$ is a refined version of the $TAC^0$ in Clote and Takeuti [5]. In fact, although Clote And Takeuti did not observe this, it is reasonably easy to see that $TAC^0$ cannot prove $\mathsf{NP} = \mathsf{co\text{-}NP}$. This is because Clote and Takeuti showed the $\Delta_1^b$-predicates of $TAC^0$ are precisely $\mathsf{AC}^0$, on the other hand, it is obvious that the $\Delta_2^b$-predicates of $TAC^0$ contain $\Sigma_1^b$, the arithmetization of $\mathsf{NP}$, and so can do parity. If $TAC^0$ could prove $\mathsf{NP} = \mathsf{co\text{-}NP}$ then $\Delta_1^b = \Delta_2^b$ in $TAC^0$ giving a contradiction. Our argument to show $TAC^0$ cannot prove the collapse of the polynomial hierarchy is along the same lines as Pollett [13]; however, we feel the theories in this paper can be more easily modified (for instance, by adding modular gates) to create new theories that are powerful enough to reason about classes for which people are interested in lower bounds but not strong enough to prove the collapse of the hierarchy.

Also, unlike the case of Pollett [13] we argue there is interesting mathematics that can be carried out in $TAC^0$. To support our claim we prove three results in these theories. We show there is a very simple universal predicate for $\Sigma_k^{\log}$ which *LIOpen* can reason about. We use this to show *LIOpen* proves $\Sigma_k^{m\cdot\log} \neq \Pi_k^{m\cdot\log}$ for every $m$, and $k$. This implies *LIOpen* proves that $\mathsf{mLH}$ is infinite. Our second result is that $TAC^0$ proves the $\mathsf{LH}$ is not equal to $\Sigma_m$-$\mathsf{TIME}(s)$ for any $m > 0$ where $s \in L$, $s \in \omega(\log n)$. In particular, this shows $TAC^0$ can prove $\mathsf{LH} \subsetneq \mathsf{NP}$. Our third result is that $TAC^0$ proves $\mathsf{LH} \neq \mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$ for all $k$ and $m$. The right hand class allows a slightly larger than log-sized number of existential queries to be made in an otherwise $\Sigma_k$-$\mathsf{LOGTIME}$ machine. This suggests that $TAC^0$ might be able to prove the log-hierarchy is infinite although the argument would most likely be quite different than that of Hastad [6].

The remainder of this paper is organized as follows: In the next section, we present the notations and bounded arithmetic theories we will be discussing in this paper. In Section 3, we characterize the $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-definable functions of $(T\Sigma_k^{\log})'$. Then in the next section, we show the three separations we can prove in $(T\Sigma_k^{\log})'$ and $TAC^0$. Finally, in the last section we show neither $TAC^0$ nor $ZAC^0$ proves the collapse of the polynomial time hierarchy.

# 2 Preliminaries

## 2.1 The Logtime Hierarchy

We begin by specifying the type of machines we use to define the log-time hierarchy. Our machines will be alternating $m$-tape Turing machines. If an ATM is being used to compute an $m'$-ary predicate ($m' < m$), then $m'$ of these tapes will be input tapes which are read-only. We assume the halting states of our machines are partitioned into accepting and rejecting states.

Contents of cells on any tape are read via a query mechanism that works as follows: A number of the states of the machine are designated as query states, and each of these is associated with two of the machine's tapes. When the machine enters a query state associated with tapes $j$ and $j'$, then the subsequent action of the machine can be dependent upon whether the $i$th bit of tape $j'$ is a 1 or a 0, where $i$ is the contents of tape $j$. The contents of the tapes are not altered by this procedure.

A language is in $\mathsf{DLOGTIME} = \Sigma_0$-$\mathsf{LOGTIME}$=$\Pi_0$-$\mathsf{LOGTIME}$ if it is recognized by an ATM of the above type in log-time using only deterministic states (including input query states). A language is $\Sigma_{k+1}$- (resp. $\Pi_{k+1}$) - $\mathsf{LOGTIME}$ if it can be recognized by an ATM of the above type that begins in an existential (universal) state and makes at most $k$-alternations between existential and universal states along any branch. The log-hierarchy is defined by $\mathsf{LH} := \cup_k \Sigma_k$-$\mathsf{LOGTIME}$ $= \cup_k \Pi_k$-$\mathsf{LOGTIME}$. We write $\mathsf{B}(\Sigma_k$-$\mathsf{LOGTIME})$ for predicates that are a Boolean combination of $\Sigma_k$-$\mathsf{LOGTIME}$ predicates. We say a poly-sized function $f(x)$ is computed by an ATM of one of the above types iff $\mathrm{BIT}(i, f(x))$ = the $i$th bit of $f(x)$ can be computed by a single such machine with inputs $i$ and $x$. For this paper we will need to work in the language $L$ consisting of the symbols 0, 1, $\leq_l$, $|x|$, PAD, CAT, MSP, LSP, and $\#_l$. The symbols 0, 1 are intended to have their usual meaning; the intended meanings of the remaining symbols are: $|x| := \lceil \log_2(x+1) \rceil$, $x \leq_l y := |x| \leq |y|$, $x =_l y := |x| = |y|$, $\mathrm{PAD}(x, y) := x \cdot 2^{|y|}$, $\mathrm{CAT}(x, y) := x \cdot 2^{|y|} + y$ $(= \mathrm{PAD}(x, y) + y)$, $\mathrm{MSP}(x, i) := \lfloor x/2^i \rfloor$, $\mathrm{LSP}(x, i) := x - \lfloor x/2^i \rfloor \cdot 2^i$, and $x \#_l y := 2^{2^{||x||+||y||}}$.

The terms in $L$ can be combined to form a large number of interesting and useful terms. Below are some abbreviations we will frequently use for $L$-terms.

| | |
|---|---|
| $2 \cdot x := \mathrm{PAD}(x, 1)$ | $\|x\| + 1 := \|\mathrm{CAT}(x, 1)\|$ |
| $\|x\| \dot{-} 1 := \|\mathrm{MSP}(x, 1)\|$ | $\|x\| \dot{-} a := \|\mathrm{MSP}(x, a)\|$ |
| $2^{\|x\|} := \mathrm{PAD}(1, x)$ | $2^{\min(\|x\|, a)} := \mathrm{MSP}(2^{\|x\|}, \|x\| \dot{-} a)$ |
| $a +_x b := \|\mathrm{CAT}(2^{\min(\|x\|,a)}, 2^{\min(\|x\|,b)})\| \dot{-} 2$ | $a \dot{-}_x b := \|\mathrm{MSP}(2^{\min(\|x\|,a)}, b)\| \dot{-} 1$ |
| $a \leq_x b := a \dot{-}_x b =_l 0$ | $a =_x b := a \leq_x b \wedge b \leq_x a$ |
| $\lfloor \frac{1}{2} x \rfloor := \mathrm{MSP}(x, 1)$ | $2^{\lfloor \frac{1}{2}\|x\| \rfloor} := \mathrm{PAD}(1, \lfloor \frac{1}{2}\|x\| \rfloor)$ |
| $2^{\|x_1\|+\|x_2\|} := \mathrm{PAD}(\mathrm{PAD}(1, x_1), x_2)$ | $2^{\sum_{i=1}^{n} \|x_i\|} := \mathrm{PAD}(2^{\sum_{i=1}^{n-1} \|x_i\|}, x_n)$ |
| $\beta_t(i, w) := \mathrm{LSP}(\mathrm{MSP}(w, \mathrm{PAD}(i, \|t\|)), 2^{\|\|t\|\|})$ | $\mathrm{BIT}(i, w) := \beta_0(i, w)$ |
| $\mathrm{cons}(x, y) := \mathrm{CAT}(\mathrm{PAD}(\mathrm{CAT}(2^{\|y\|}, x), 2^{\|x\|}), y)$ | $\langle x \rangle := \mathrm{cons}(x, 0)$ |
| $\mathrm{car}(w) := \mathrm{LSP}(\mathrm{MSP}(w, \lfloor \frac{1}{2}\|w\| \rfloor, \lfloor \frac{1}{2}\|w\| \rfloor \dot{-} 1)$ | $\mathrm{cdr} w := \mathrm{LSP}(w, \lfloor \frac{1}{2}\|w\| \rfloor)$ |

$(w)_0 := \mathrm{car}(w)$ $\quad$ $(w)_n := \mathrm{car}(\mathrm{cdr}^{(n)}(w))$

$\mathrm{bool}(x) := \mathrm{MSP}(x, |x| \dot{-} 1)$ $\quad$ $\mathrm{lbool}(x, y) := \mathrm{PAD}(\mathrm{bool}(x), |y|)$

$\mathrm{K}_{\leq_l}(a, b) := 1 \dot{-} (|a| \dot{-} |b|)$ $\quad$ $\mathrm{K}_{\wedge}(a, b) := \mathrm{cond}(a, \mathrm{cond}(b, 1, 0), 0)$

$\mathrm{K}_{=_l}(a, b) := \mathrm{K}_{\wedge}(\mathrm{K}_{\leq_l}(a, b), \mathrm{K}_{\leq_l}(b, a))$ $\quad$ $\mathrm{K}_{\neg}(a) := \mathrm{cond}(a, 0, 1)$

$\mathrm{K}_{\leq_x}(a, b) := \mathrm{K}_{=_l}(a \dot{-}_x b, 0)$ $\quad$ $\mathrm{K}_{=_x}(a, b) := \mathrm{K}_{\wedge}(\mathrm{K}_{\leq_x}(a, b), \mathrm{K}_{\leq_x}(b, a))$

$\#_l^0(x) := x$ $\quad$ $\#_l^{k+1}(x) := x\#_l(\#_l^k x)$

$$\begin{aligned} \langle x_1, \ldots x_n \rangle &:= \mathrm{cons}(x_1, \langle x_2, \ldots, x_n \rangle) \\ \mathrm{cond}(x, y, z) &:= \mathrm{LSP}(\mathrm{MSP}(\mathrm{CAT}(y, z), \mathrm{lbool}(x, z)), \mathrm{lbool}(1 \dot{-} |x|, z)) \end{aligned}$$

Note that operations involving $a$ and $b$ work as expected provided $|x| \geq a$ and $|x| \geq b$, so $a$ and $b$ should be thought of as “small numbers”. The subscripts on $+_x$ and $\dot{-}_x$ will be dropped when it is clear that the $a$ and $b$ involved are small enough to easily build a suitable $x$. By repeating $x_i$'s in the above we can make a term $2^{\sum_i a_i |x_i|}$ for any fixed integers $a_i$. $\beta$ allows block sequence coding. Roughly, $\beta_t(i, w)$ projects out the $i$th block (starting with a 0th block) of $2^{||t||}$ bits from $w$. $\langle x, y \rangle$ is a pairing function and $(\langle x, y \rangle)_0 = x$, $(\langle x, y \rangle)_1 = y$. $\mathrm{bool}(()x)$ is 1 if $x > 0$ and 0 otherwise; $\mathrm{lbool}(()x, y)$ is $2^{|x|}$ if $x > 1$ and 0 otherwise; and $\mathrm{cond}(x, y, z)$ returns $y$ if $x > 0$ and $z$ otherwise.

We syntactically enlarge first order logic to include *bounded quantifiers* of the form $(\forall x \leq_l t)$ and $(\exists x \leq_l t)$ with $x$ not occurring in $t$. Since $x \leq_l t$ means $|x| \leq |t|$, a quantifier $(\forall x \leq_l t)$ should be interpreted as meaning $(\forall x)(x \leq 2^{|t|} \supset \cdots)$ and one of the form $(\exists x \leq_l t)$ should be interpreted as meaning $(\exists x)(x \leq 2^{|t|} \wedge \cdots)$. A quantifier is called *sharply bounded* if the bounding term $t$ is of the form $|s|$ for some term $s$. A formula is called (sharply) bounded if all quantifiers in it are (sharply) bounded. Generalizing, given a term $\ell$, a quantifier is $\ell$-bounded if the bounding term is of the form $\ell(s(x))$.

Let $\tau$ be a set of nondecreasing 1-ary $L$-terms. We write $|\tau|$ for the set of terms $|\ell|$ where $\ell \in \tau$. We will be interested in sets $\tau$ with the following closure property: if $\ell_1, \ell_2 \in \tau$ and $s, t \in L$ then there is a term $\ell' \in \tau$ and $r \in L$ such $\mathrm{CAT}(\ell_2(s), \ell_1(t)) \leq_l \ell'(r)$. In this case we will say that $\tau$ is weakly CAT closed. A predicate $\phi(x)$ is in $\Sigma_k$-$\mathsf{TIME}(|\tau|)$ if it can be computed by a $\Sigma_k$-ATM in time $|\ell'(x)|$ for some $\ell' \in \tau$. If $\tau$ is weakly CAT closed, then the time classes we work with will be closed under “multiplication by constants,” e.g., if $\tau = \{id\}$ contains only the identity term, then running times bounded by any of $2 \log n$, $3 \log n$, … are allowed for $\Sigma_k$-$\mathsf{TIME}(|\tau|)$ predicates.

Next we would like to give an arithmetization of the classes $\Sigma_k$-$\mathsf{TIME}(|\tau|)$. Roughly, this can be done by appropriately bounding quantifiers and number of quantifier alternations for formulas in the language $L$. To ensure the arithmetized classes correspond to the machine classes for $k > 0$ even in the log-time case, care must be taken in how the $k = 0$ case is defined. In particular, in the log-time case we would like the $k = 0$ formulas to be in $\mathsf{DLOGTIME}$. To begin we write $\mathsf{E}_\tau\Psi$ (resp. $\mathsf{U}_\tau\Psi$) to denote formulas of the form $(\exists x \leq_l \ell(t))\phi$ (resp. $(\forall x \leq_l \ell(t))\phi$) where $\ell \in \tau$ and $\phi \in \Psi$. We write $\mathsf{E}\Psi$ (resp. $\mathsf{U}\Psi$) for $\mathsf{E}_{\{id\}}\Psi$ (resp. $\mathsf{U}_{\{id\}}\Psi$).

*open* is the class of formulas without quantifiers. If $\ell$ is a 1-ary term, then a variable $x$ in a term $f$ is said to be *$\ell$-bounded* if either (1) $x$ does not appear in $f$, or (2) $x$ appears in $f$ as $\ell(x)$, or (3) $x$ appears in $f$ as $\beta_{\ell(x)}(t, w)$, where $w$ is $\ell$-bounded in $t$. (Recall that $\beta$ was defined above to do block sequence coding.)

**Definition 1** *The $\tau$-bounded arithmetic hierarchy is defined as follows:*

1. $\Sigma_0^\tau$ *consists of all disjunctions of formulas of the form*

$$(\exists i \leq_l \mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||))\phi$$

*such that either*

(a) *$\phi$ is open, $\ell \in \tau$, and all variables (except for $i$) in $\phi$ are $|\ell|$-bounded, or*

(b) *$\phi$ is of the form*

$$(\exists j \leq_l \mathrm{MSP}(||\ell(\#_l^m(x))||, |\ell(\#_l^m(x))|_4)) \\ \phi'(\beta_{||\ell||}(t(i), w), \vec{a}) ,$$

*where $\ell \in \tau$, $t$ is $|\ell|$-bounded, $\phi'(b, \vec{a})$ is open and all variables (except for $i$) in $\phi'$ are $||\ell||$-bounded.*
*Notice $|\mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||)| \leq ||\ell||$, so $\beta_{|\ell|}(0, i) = i$.*

2. $\Pi_0^\tau$ *consists of all formulas of the following form*

$$(\forall i \leq_l \mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||))\phi$$

*such that*

(a) *$\phi$ is open, $\ell \in \tau$, and all variables (except for $i$) in $\phi$ are $|\ell|$-bounded, or* *(b) $\phi$ is of the form:*

$$(\forall j \leq_l \mathrm{MSP}(||\ell(\#_l^m(x))||, |\ell(\#_l^m(x))|_4)) \\ \phi'(\beta_{||\ell||}(t(i), w), \vec{a}) \ ,$$

*where $\ell \in \tau$, $t$ is $|\ell|$-bounded, $\phi'(b, \vec{a})$ is open and all variables (except for $i$) in $\phi'$ are $||\ell||$-bounded.*

3. *$\Delta_1^\tau$ are boolean combinations of open and $\Sigma_0^\tau$-formulas.*

4. *$\Sigma_1^\tau$ is the class $\mathsf{E}_\tau\Delta_1^\tau$. $\Pi_1^\tau$ is the class $\mathsf{U}_\tau\Delta_1^\tau$.*

5. *$\Sigma_k^\tau$ is the class $\mathsf{E}_\tau\Pi_{k-1}^\tau$. $\Pi_k^\tau$ is the class $\mathsf{U}_\tau\Sigma_{k-1}^\tau$.*

We write $\Sigma_k^{\log}$ and $\Pi_k^{\log}$ for $\Sigma_k^{\{|id|\}}$ and $\Pi_k^{\{|id|\}}$. Similarly, we write $\Sigma_i^{\mathsf{b}}$ and $\Pi_i^{\mathsf{b}}$ for $\Sigma_i^{\{id\}}$ and $\Pi_i^{\{id\}}$. A predicate is in $\check{\Sigma}_k^\tau$ (resp. $\check{\Pi}_k^\tau$) if its $\Delta_1^\tau$-subformula is actually in $\Sigma_0^\tau$ or $\Pi_0^\tau$. We will show in a moment that $\Sigma_k^{\log}$ and $\check{\Sigma}_k^{\log}$ predicates correspond to $\Sigma_k$-$\mathsf{LOGTIME}$ and $\Sigma_i^{\mathsf{b}}$ predicates correspond to $\Sigma_i^{\mathsf{p}}$ and $\Pi_i^{\mathsf{p}}$. We will use the more general definitions in the section where we talk about the power of $TAC^0$.

For any class of formulas $\Psi$ we write $\mathsf{B}(\Psi)$ to denote Boolean combinations of formulas in $\Psi$. We write $\Sigma_k^\tau(\Psi)$ (resp. $\Pi_k^\tau(\Psi)$) for the class of formulas which would be $\Sigma_k^\tau$-formulas (resp. $\Pi_k^\tau$-formulas) if we treated all $\Psi$ subformulas as atomic. Finally, a formula $B$ is in $\mathsf{L}\Psi$ if there is a formula $A \in \Psi$ of which $B$ is a subformula. As an example of using these definitions and the $\mathsf{E}_\tau$ notation from before, consider the expression $\mathsf{LE}_{\{||id||\}}\mathsf{B}(\Sigma_k^{\log})$. This is the class of subformulas of $\mathsf{E}_{\{||id||\}}\mathsf{B}(\Sigma_k^{\log})$ formulas. These in turn are formulas consisting of a quantifier of the form $(\exists x \leq_l ||t||)$ for some term $t$ followed by a Boolean combination of $\Sigma_k^{\log}$-formulas.

Given our above abbreviations we define a last set of hierarchies. A predicate $A(x_1, \cdots, x_n)$ is in $\Sigma_k^{m\cdot\log}$ (resp. $\Pi_k^{m\cdot\log}$) where $m$ is a constant if it is in $\check{\Sigma}_k^{\log}$ (resp. $\check{\Pi}_k^{\log}$) and all terms in it sharply bounded quantifiers are bounded by $|\#_l^m(2^{\sum_{i=1}^n |x_i|})|$ and the term in its innermost quantifier is bounded by
$||\#_l^m(2^{\sum_{i=1}^n |x_i|})||$. We write $\mathsf{mLH}$ for $\cup_k(\Sigma_k^{m\cdot\log} \cup \Pi_k^{m\cdot\log})$.

**Lemma 1** *If $t$ is an $L$-term, then in* $\mathsf{DLOGTIME}$ *a Turing Machine can (1) write $|t|$ on a blank tape and (2) compute* $\mathrm{BIT}(i, t)$*. Hence, the open formulas in the language $L$ can be evaluated in* $\mathsf{DLOGTIME}$.

**Corollary 1** *Let $\tau$ be weakly* CAT *closed and also only contain terms which are $\Omega(|x|)$. Then (1) the $\mathsf{B}(\Sigma_0^{|\tau|})$-predicates are in $\mathsf{DTIME}(|\tau|)$ and (2) the $\mathsf{B}(\Sigma_0^{\{|id|\}})$-predicates are in $\mathsf{DTIME}(|id|) = \mathsf{DLOGTIME}$.*

*Proof.* (Of Lemma 1) We prove both statements of the lemma by simultaneous induction on the complexity of the term $t$. If $t$ is 0 or 1, the result is obvious. If $t = x$ (a variable), then $|t| = |x|$ can be computed in $\mathsf{DLOGTIME}$ as follows: Using a blank tape to hold the query string, first query the bits 1, 10, 100, ... of $x$ until the first time one finds a blank symbol; then erase the last zero and return the tape head to the left; finally, make one more pass left to right changing 0's to 1's whenever querying the resulting bit position does not yield a blank. This will leave $|x|$ written on the query tape. $\mathrm{BIT}(i, x)$ can be computed using the query mechanism of our Turing machines.

Now suppose that $t$ is $\mathrm{CAT}(u, v)$, where $u$ and $v$ are terms. By induction we can compute $|u|$ and $|v|$ on separate tapes. $|t| = |u| + |v|$ can be computed by the standard addition algorithm in $\mathsf{DLOGTIME}$. To compute the $i$th bit of $t$, we first use the standard subtraction algorithm to calculate $j = i \dot{-} |u|$. Then we query the $i$th bit of $u$ or the $j$th bit of $v$, depending on whether $i < |u|$ or $i \geq |u|$.

Similar arguments can be used in the cases where the $t$ is made from other terms using PAD, MSP, LSP, or $\#_l$. That predicates can be evaluated in $\mathsf{DLOGTIME}$ is clear from the fact that they can be expressed as terms that evaluate to 0 or 1 using $\mathrm{K}_{\leq_l}$, etc. $\Box$

*Proof.* (Corollary 1) The $\mathsf{DTIME}(|\tau|)$ predicates are closed under AND and NOT, so it suffices to show

$$(\exists i \leq_l \mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||))\ \phi(i, x)$$

is in $\mathsf{DTIME}(|\tau|)$ where $\phi$ is open and variables in $\phi$ are $|\ell|$-bounded for some $\ell \in \tau$. The universal form of $\Pi_0^\tau$-predicate is proved similarly. For ease of notation, we are assuming that the only variables occuring in $\phi$ are $i$ and $x$.

We first write

$$2^{|\mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||)|}$$

on a blank tape in $\mathsf{DLOGTIME} \subseteq \mathsf{DTIME}(|\tau|)$. This can be done because the length of this term is $O(||\ell(x)||)$ for some $\ell \in \Omega(|x|)$ and each bit can be computed in $O(||\ell(x)||)$ time. In a combined time not exceding $O(|\ell(x)|)$ we can count backwards from this value to 0. This will allow us to iterate through the necessary values of $i$.

The number of values of $i$ such that $i \leq_l \mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||)$ is $O(|\ell(x)|/||\ell(x)||)$, so if we can show that for each $i$, $\phi$ can be evaluated in $O(||\ell(x)||)$ time, then we can evaluate $\phi$ for all values of $i$ in $O(||\ell(x)|| \cdot |\ell(x)|/||\ell(x)||) = O(|\ell(x)|)$ time.

In Definition 1, $\phi$ is either open or has an additional existential quantifier like the above but with one more length nesting. We describe how to handle the open case, the second case can be handle by repeating the argument of the last paragraph with the additional length nesting followed by the open case. By Lemma 1, the length of any term $t$ and any bit of a term $t$ can be computed in $\mathsf{DLOGTIME}$ of its inputs. Consider the terms that may appear in $\phi$. If we have a term like $|\ell(x)|$ we can write it on a new tape in time $O(||\ell(x)||)$ time, and then to access any bit of this number it takes time $O(||\ell(x)||)$.

Similarly, if we have a term like $\beta_{|\ell(x)|}(t, w)$ and we assume for each $i$ after some initial preprocessing $t$ is computable in time $O(||\ell(x)||)$, then we can write out the $||\ell(x)||$ bits of $\beta_{|\ell(x)|}(t, w)$ in $O(||\ell(x)||)$ time. Operations involving $i$ can be computed in log-time of $|i|$, that is, in time $O(|||\ell(x)|||)$. So for each $i$, we can compute $\phi(i, x)$ in $\mathsf{DTIME}(||\ell||)$. $\Box$

**Lemma 2** *Let $\tau$ be weakly* CAT *closed and also only contain terms which are $\Omega(\log n)$. For $k \geq 1$, both the $\Sigma_k^\tau$ and $\check{\Sigma}_k^\tau$ (resp. $\Pi_k^\tau$ and $\check{\Pi}_k^\tau$) predicates define the same sets of natural numbers as $\Sigma_k$-$\mathsf{TIME}(|\tau|)$ (resp. $\Pi_k$-$\mathsf{TIME}(|\tau|)$) where $|\tau|$ is $|\ell|$ for $\ell \in \tau$. In particular, this implies $\Sigma_i^{\mathsf{b}}$-formulas define the same sets of natural numbers as $\Sigma_i^{\mathsf{p}}$-predicates and for $k \geq 1$, $\Sigma_k^{\log}$-formulas define the same sets of natural numbers as $\Sigma_k$-$\mathsf{LOGTIME}$.*

*Proof.* We will sketch the $\Sigma_1^{\log}$ case. It is straightforward to then generalize this to $k > 1$ and $\tau$ different from $\{|id|\}$. If $\phi$ is in $\Sigma_1^{\log}$ then we can construct a machine $M_\phi$ which in a sequence of existential moves guesses the outermost existential quantifier of $\phi$ and then uses this value to compute the $\Delta_1^{\log}$ part of the formula. Since $\Delta_1^{\log}$ is in $\mathsf{DLOGTIME}$ by Lemma 1, $\check{\Sigma}_1^{\log} \subseteq \Sigma_1^{\log} \subseteq \Sigma_1$-$\mathsf{LOGTIME}$.

For the other direction, let $M$ be a $\Sigma_1$-$\mathsf{LOGTIME}$ ATM with $m$ tapes. We need to show that the predicate accepted by $M$ can be represented as a $\check{\Sigma}_1^{\log}$ predicate. We do this by introducing a suitable encoding scheme for the computation of $M$. Unfortunately, a tableau for (a path of) such a computation is in general of size $O(\log^2(n))$ and so may be too large. Our strategy will be to write down a logarithmically-sized "outline" $w$ of such a tableau (much of which contains redundant information anyway), and a nondeterminism path $p$, from which we can still carry out the local verifications necessary to tell the results of the computation (along the path $p$).

Choose a constant $K$ (dependent on the machine $M$ but not on the input $x$) such that

1. $K = 2^c$ is a power of 2;
2. $\frac{K^2}{4}\log(|x|)$ bounds the running time of $M$ on all paths and all inputs of length $n$; and
3. $K$ is larger than the product of the number of states of $M$,
4. $K$ is more than twice the number of tape symbols (including blanks) in the alphabet of $M$.

In particular, this means that we can code instructions and tape symbols (tagged to indicate whether or not the head is reading the symbol) using numbers between 0 and $K$, all of which have length at most $c$. We can approximate $K\frac{\log|x|}{\log\log|x|}$ by the $L$-term $T_x = \mathrm{MSP}(\mathrm{PAD}(\mathrm{MSP}(||x||, |x|_4), K), 1)$. We write $B_x$ for the term $K2^{|x|_4}$.

Notice the following useful facts about $B_x$:

1. $B_x$ is a power of 2, so $2^{|B_x|} = 2B_x$ and $rB_x = \mathrm{MSP}(\mathrm{PAD}(r, B_x), 1)$.
2. $B_xT_x \in O(\log n)$.
3. The running time of $M$ on input $x$ is bounded by $R_x := B_xT_x$.

From now on we will write $B$ for $B_x$, $T$ for $T_x$, and $R$ for $R_x$. For our abbreviated tableau encoding of a computation path of $M$, we will divide both the steps of the computation of $M$ and the cells of the tapes of $M$ into $T \in O(\frac{\log|x|}{\log\log|x|})$ blocks of length $B \in O(\log\log|x|)$ each. We will use $\beta_z(r, w)$ to access the $r$th block of length $|z|$ out of $w$. Our encoding of (a path of) the computation consists of a bit string $\mathrm{cons}(w, p)$, where $w$ and $p$ are made up of the following pieces:

1. The first $cR$ bits of $w$ code via block coding ($R$ blocks of $c$ bits per block) the state the machine is in at each time $t \in [0, R)$.

   Let

   $$\mathrm{INST}(t) := \beta_K(t, w)$$

   denote the code for the instruction executed at time $t$. We will call these bits the *instruction bits* of $w$.

2. The next $mR$ bits ($mT$ blocks of $B$ bits) of $w$ code the index of the block in which the $m'$th tape head is located at each time $rB$ for $r \in [0, T)$. Notice $|r| \leq |T| \leq B$, so this requires at most $B$ bits to encode. Let

$$\text{BLOCK}(m', r) := \beta_{2^B}(m'T + r + cT, w)$$

denote the index of the block in which the head of tape $m'$ is located at time $rB$. Here, and in the formulas below, '$+$' is really '$+_{2^R}$' but we drop the subscript for readability. We are adding $cT$ to move past the initial $cR = cTB$ bits of $w$ that code the state of the machine at each step.

3. The next $3mT$ blocks of $w$ code for each $r \in [0, R)$ the contents (tagged to indicate tape head presence) of the $3B$ tape cells that include the block where the $m'$th tape head is located at time $rB$ and the block to the left and right of this block. The code for the block at time $rB$ is given by

$$\text{BCONT}(m', r, \alpha) := \beta_{2^B}(3m'T + \alpha T + r + mT + cT, w)$$

where $\alpha = 0, 1, 2$ are the codes for the left, middle, and right blocks respectively. Notice we are first coding all the left blocks, then the blocks where the tape head is located, then the right blocks.

4. The next $mT$ blocks of $w$ code for each $r \in [0, R)$ and each $m' \in [0, m)$ the relative position of the head on tape $m'$ within these 3 blocks (left, middle, right) at time $rB$.

$$\text{POS}(m', r) := \beta_{2^B}(m'T + r + 4mT + cT, w) \ .$$

is a number of length at most $|3B| \leq B$ so this is a somewhat wasteful encoding.

Note that between time $rB$ and $(r + 1)B$ the head on any tape only has time to move in the current tape block and at most one of the blocks immediately to the left or to the right, but not both.

5. The next $3mT$ blocks of $w$ code for each $\alpha \in \{0, 1, 2\}$ (interpretted as in BCONT ) and each $r \in [0, T)$, the index of the last time block prior to the $r$th time block during which the $\alpha$ part of tape block BLOCK$(m', r)$ was one of the three subblocks being considered (or 0 if never). So we let

$$\text{PREV}(m', r, \alpha) := \beta_{2^B}(3m'T + \alpha T + r + 5mT + cT, w) \ .$$

6. The next $3mT$ blocks of $w$ code in an analogous fashion to the above the index of the time block of the next visit (or $B-1$ if none). We define the term

$$\text{NEXT}(m', r, \alpha) := \beta_{2^B}(3m'T + \alpha T + r + 8mT + cT, w) .$$

in the same fashion as PREV.

7. The next $11mR$ bits code for each $r \in [0, T)$ and $s \in [0, B/\log B)$ (and $\alpha$ as necessary) the log-scaled down versions of items 2 – 6 That is, we break down the time period between time $rB$ and $(r+1)B$ into $B/\log B$ blocks of size $\log B$ and record the contents of the sub-block of size $\log B$ on each tape containing the head (and its left and right neighboring subblocks) and then the next, and previous sub-blocks for each of these. We can define, in anology to BLOCK$(m', r)$, PREV$(m', r, \alpha)$, NEXT$(m', r, \alpha)$, BCONT$(m', r, \alpha)$ and to POS$(m', r)$, SUBBLOCK$(m', r, s)$, SUBPREV$(m', r, s, \alpha)$, SUBNEXT$(m', r, s, \alpha)$, SUBBCONT$(m', r, s, \alpha)$ and SUBPOS$(m', r, s)$. (There is no need to code a log-scaled version of INST.)

8. The last $2^{4mK \log B}(2^{4mK \log B^2})$ bits (which we will call the LOOKUP bits) code for each of the possible configurations of $3K \log B$ bits on each of the $m$ tapes as well as their head position, how $M$ would evolve for $\log B$ steps. Notice this number of bits is less than $2^{4mK \log B^3}$. Since $B \in O(|x|_3)$, this number is less than $||x||/|||x|||$.

9. The encoding above will require a total of $O(\log |x|)$ bits. To this we add the string $p$ containing the at most $R \in O(\log |x|)$ nondeterministic guesses made by $M$. (Note, we may assume that $M$ always has exactly two choices.) We let GUESS$(t, p) :=$ BIT$(t, p)$.

Now we need to construct a predicate $\varphi(x)$ that asserts that machine $M$ accepts the input $x$. $\varphi(x)$ will be a formula $(\exists w \leq_l |s|)\psi(x, w)$ where $\psi$ is a $\Pi_0^{\log}$-formula asserting $w$ codes a computation of $M$ on $x$ followed by a string of nondeterministic guesses used on input $x$. $\psi$ consists of the conjunction of the following:

1. *Blank tapes at start. Computation begins in start state.* What we really require here is that whenever a block is first accessed its tapes squares are blank. A block is said to be accessed for the first time if its previous pointer is 0. We use 0 as our code for blank and assume

without log of generality that the starting state of $M$ is state 0, so this check becomes just:

$$\beta_K(0, \text{INST}) = 0 \land \\ (\forall r \leq_l T) \bigwedge_{m'} \bigwedge_\alpha \text{PREV}(m', r, \alpha) =_l 0 \supset \text{BCONT}(m', r, \alpha) =_l 0$$

Here $\land_{m'}$ and $\land_\alpha$ are just finite conjunctions over the values of $m'$ and $\alpha$. Given our definition of PREV and BCONT this check can be seen to meet the definition of $\Pi_0^{\log}$.

2. *Proper initial head location.* Tape head is at left of all tapes at start of computation.

$$\bigwedge_{m'} \text{POS}(m', 0) =_l 0 \ .$$

Given the definition of POS this will be a $\Pi_0^{\log}$ predicate if we add a trivial dummy quantifier.

3. *Tape contents only change by action of the machine.* If $M$ leaves the vicinity of a tape block and returns later, the contents of the tape block should be the same when it returns as they were when it left.

$$(\forall r \leq_l T) \bigwedge_{m'} \bigwedge_\alpha [\text{PREV}(m', \text{NEXT}(m', r, \alpha), \alpha) = r \\ \land \neg\text{NEXT}(m', r, \alpha) = r + 1 \supset \\ \text{BCONT}(m', \text{BLOCK}(m', \text{NEXT}(m', r, \alpha)), \alpha) = \text{BCONT}(m', r, \alpha).$$

4. *Move at most one tape block each time block.* For each tape $m'$, if the block at time $rB$ is $t$, then the block at time $(r+1)B$ is $t$, $t-1$ or $t+1$. This can be expressed as:

$$(\forall r \leq_l T)(\text{BLOCK}(m', r) = \text{BLOCK}(m', r+1) \\ \lor \text{BLOCK}(m', r) = \text{BLOCK}(m', r+1) \dot{-} 1 \\ \lor \text{BLOCK}(m', r) = \text{BLOCK}(m', r+1) + 1).$$

We are dropping the subscripts on $+$, $\dot{-}$, and $=$ for readability.

5. *Main information is consistent with log-scaled information.* For each $r \in [0, T)$, tape $m'$, and $s \in [0, \log B)$, the information in SUBBCONT, SUBPREV, SUBNEXT, SUBPOS, SUBBLOCK must be consistent with $\text{BCONT}(r, m', \alpha)$ and yield $\text{BCONT}(r+1, m', \alpha)$.

This can be checked with a formula of the second type in the definition of $\Pi_0^{\log}$ predicate, i.e., by a formula $\forall r \leq T\ \forall s \leq \log B\ \theta$, where $\theta$ is a conjunction of the following:

(a) *Initial locations consistent.*

- $\text{SUBBLOCK}(m', r, 0) = \text{MSP}(\text{POS}(m', r), ||B||)$.
- $\text{SUBPOS}(m', r, 0) = \text{LSP}(\text{POS}(m', r), |B|)$.

(b) *Initial contents consistent.*

- $\text{SUBPREV}(m', r, s, \alpha) = 0 \wedge s < |B| \supset$ $\text{SUBBCONT}(m', r, s, \alpha) = \beta_B(s, \text{BCONT}(m', r, 0))$.
- $\text{SUBPREV}(m', r, s, \alpha) = 0 \wedge |B| \leq s < |B| \supset$ $\text{SUBBCONT}(m', r, s, \alpha) = \beta_B(s - |B|, \text{BCONT}(m', r, 1))$.
- $\text{SUBPREV}(m', r, s, \alpha) = 0 \wedge 2|B| \leq s \supset$ $\text{SUBBCONT}(m', r, s, \alpha) = \beta_B(s - 2|B|, \text{BCONT}(m', r, 2))$.

(c) *Final contents consistent.* Similar to 5(b).

6. *The log-scaled information is consistent with the* LOOKUP *bits.* This amounts to checking that $\text{SUBBCONT}(r+1, s, m', \alpha)$ agrees with what you get by looking up $\text{SUBBCONT}(r, s, m', \alpha)$ in the LOOKUP bits and seeing what the resulting blocks would be after $\log B$ steps. Again this can be check with the second quantifier type of $\Pi_0^{\log}$ predicate.

7. *The* LOOKUP *bits are consistent with the behavior of the machine $M$.* For each possible $3K \log B$ bits on each of the $m$ tapes as well as their head position, that the next $\log B$ blocks of size $4K \log B$ correctly represent how $M$ would evolve. Since the number of things we have to check is less than $T$ we can use the first quantifier type of $\Pi_0^{\log}$ predicate to check this.

   Again by the same arguments as for BLOCK this will be DLOGTIME computable.

8. *Halting configuration.* Check the last state recorded in the INST bits is an accepting state. This can be done by a simple projection.

Since each of these checks can be put in the form of a $\Pi_0^{\log}$-predicate, $M$ can be computed by $\Sigma_1^{\log}$-predicate. □

## 2.2 *BASIC* and other Bounded Arithmetic Theories

We now introduce some arithmetic theories, beginning with *BASIC*. The version of *BASIC* presented below is inspired by the *BASIC* of Buss [3] but where we have modified our axioms to the symbols of our language.

**Definition 2** *Recall that* $x =_l y$ *is an abbreviation for* $x \leq_l y \wedge y \leq_l x$. *The binary predicate* $x = y$ *is an abbreviation for the formula*

$$a =_l b \wedge (\forall i \leq_l |a|)(\mathrm{BIT}(i, a) =_l \mathrm{BIT}(i, b)).$$

*Equality axioms are axioms of the form*

$$t = s \supset f(t) = f(s) \text{ or}$$

$$t = s \wedge A(t) \supset A(s)$$

*where* $f$,$t$, *and* $s$ *are terms and* $A$ *is atomic.*

**Definition 3** *The theory BASIC consists of the following*

1. *All substitution instances of the following finite set of quantifier free axioms for the non-logical symbols of our language:*

| | |
|---|---|
| $\neg(0 =_l 1)$ | $\neg x =_l 0 \supset \mathrm{PAD}(x, 1) =_l \mathrm{CAT}(x, 1)$ |
| $\|0\| =_l 0$ | $y \leq_l x \vee x \leq_l y$ |
| $\|1\| =_l 1$ | $x \leq_l y \wedge y \leq_l z \supset x \leq_l z$ |
| $0 \leq_l x$ | $x \leq_l y \supset \|x\| \leq_l \|y\|$ |
| $x \leq_l \mathrm{PAD}(x, y)$ | $\neg x =_l 0 \supset x \leq_l \mathrm{PAD}(x, 1) \wedge \neg \mathrm{PAD}(x, 1) =_l x$ |
| $x \leq_l \mathrm{CAT}(x, y)$ | $x \leq_l z \wedge z \leq_l \mathrm{PAD}(x, 1) \supset x =_l z \vee z =_l \mathrm{PAD}(x, 1)$ |

2. *All substitution instances of equality axioms.*

3. *All substitution instances of the following additional axioms involving equality:*

| | |
|---|---|
| $0 = \|0\|$ | $\mathrm{PAD}(x, \mathrm{PAD}(y, z)) = \mathrm{PAD}(x, \mathrm{PAD}(z, y))$ |
| $1 = \|1\|$ | $\mathrm{PAD}(\mathrm{PAD}(x, y), z)) = \mathrm{PAD}(x, \mathrm{PAD}(y, z))$ |
| $\mathrm{MSP}(\mathrm{PAD}(x, y), \|y\|) = x$ | $\mathrm{MSP}(x, \|i\| + 1) = \mathrm{MSP}(\mathrm{MSP}(x, \|i\|), 1)$ |
| $\mathrm{LSP}(\mathrm{PAD}(x, y), \|y\|) = 0$ | $x = \mathrm{CAT}(\mathrm{MSP}(x, z), \mathrm{LSP}(x, z))$ |
| $\|x \#_l y\| = 2^{\|\|x\|\| + \|\|y\|\|}$ | $\mathrm{MSP}(\mathrm{CAT}(x, y), \|y\|) = x$ |
| $\mathrm{MSP}(x, 0) = x$ | $\mathrm{LSP}(\mathrm{CAT}(x, y), \|y\|) = y$ |
| $\mathrm{MSP}(x, \|x\|) = 0$ | $\neg y \leq_l 0 \supset \|y\| = \|\mathrm{PAD}(\mathrm{MSP}(y, 1), 1)\|$ |
| $\mathrm{LSP}(x, 0) = 0$ | $\mathrm{LSP}(x \#_l y, 1) = 0$ |
| $\mathrm{LSP}(x, \|x\|) = x$ | |

$$\mathrm{PAD}(\mathrm{MSP}(x, 1), 1) = x \vee \mathrm{CAT}(\mathrm{MSP}(x, 1), 1) = x$$

Proofs in our theories will be carried out in the sequent calculus system $LKB$ which is the usual first order sequent calculus extended with the following inferences to handle bounded quantifiers:

$$\frac{A(t),\Gamma\rightarrow\Delta}{t\leq_l s,\forall x\leq_l s A(x),\Gamma\rightarrow\Delta} \quad \frac{a\leq_l t,\Gamma\rightarrow A(a),\Delta}{\Gamma\rightarrow\forall x\leq_l t A(x),\Delta} \quad \frac{a\leq_l t,A(a),\Gamma\rightarrow\Delta}{\exists x\leq_l t A(x),\Gamma\rightarrow\Delta} \quad \frac{\Gamma\rightarrow A(t),\Delta}{t\leq_l s,\Gamma\rightarrow\exists x\leq_l s A(x),\Delta}$$

This is the same system as in Buss [3] or Krajicek [9] except wherever they used $\leq$ used we use $\leq_l$. The only initial sequents we allow are axioms and sequents of the form $A \rightarrow A$ where $A$ is atomic.

We next give some additional axiom schemes and rules we will consider.

**Definition 4** *Let $\tau$ be a set of $L$-terms.*

1. *A $\Psi$-$LIND^\tau$ axiom is an axiom $LIND^\ell_A$:*

$$A(0), (\forall x \leq_l |\ell(t(b))|)(A(x) \supset A(x +_{\ell(t)} 1)) \rightarrow A(|\ell(t(b))|) \ ,$$

   *where $t \in L$, $\ell \in \tau$, and $A \in \Psi$.*

2. *Given a formula $A(z)$, $COMP_A(t)$, is the axiom*

$$\exists y \leq_l t \, \forall z \leq_l |t| \, (\mathrm{BIT}(y, z) = 1 \iff A(z)) \ .$$

   *The $\Psi$-$COMP$ axioms are all formulas of the form $COMP_A(t)$ where $A \in \Psi$ and $t$ is a term in $L$. We write $COMP_\Psi$ for the class of formulas of the above form where $A \in \Psi$.*

As an example, let $id(a) = a$. Then $\Psi$-$LIND^{\{id\}}$ is closely related to the $LIND$ induction for $\Psi$-formulas studied in Buss [3]. Since by Parikh's theorem the exponential function is not provably total in the theories we are considering, induction up to the length of a number is potentially weaker than normal induction. Other common sets of terms are $\{|id|\}$, $\{||id||\}$ or $\{|id|_m\}$ where $|id|_0 = id$ and $|id|_m = ||id|_{m-1}|$. Sets of the form $\{|id|_m\}$ are singleton sets; however, we will also consider sets of terms such as $\{2^{k|id|_m} \mid k \in \mathbb{N}\}$, $\{2^{2^{k|id|_m}} \mid k \in \mathbb{N}\}$, or $\{2^{2^{2^{k|id|_m}}} | k \in \mathbb{N}\}$, where $m$ is a fixed integer.

**Remark 1** *Length bounded induction is sufficient to prove that the string $y$ guaranteed to exist by comprehension is also unique.*

**Definition 5** *The following theories will be of interest:*

1. *For any $i \geq 0$, $S^i_2 := BASIC + \Sigma^{\mathsf{b}}_i$-$LIND^{\{id\}}$.*

2. *$LIOpen := BASIC +$ open-$LIND^{\{id\}}$.*

3. $T\Sigma_k^{\log} := BASIC + \mathsf{B}(\Sigma_k^{\log})\text{-}LIND^{\{id\}} + \mathsf{B}(\Sigma_k^{\log})\text{-}COMP$.

4. $S_2 := \cup_i S_2^i$.

5. $TAC^0 := \cup_k T\Sigma_k^{\log}$.

6. $ZAC^0 := \cup_i ZAC_i^0$, *where* $ZAC_i^0 := TAC^0 + \Sigma_i^{\mathsf{b}}\text{-}LIND^{\{|id|_{i+2}\}}$.

We will often restrict ourselves to the deduction system $(T\Sigma_k^{\log})'$ which is a restriction of $T\Sigma_k^{\log}$ in the system $LKB$ where we only allow cuts on $\mathsf{LU}_{\{||id||\}}\mathsf{B}(\Sigma_k^{\log})$-formulas.

In the next section we will perform enough bootstrapping to argue that in an appropriate expansion of the language $L$, $S_2^i$ is conservative over both the $S_2^i$ of Buss [3] and our $S_2^i$.

**Remark 2** *The same proofs as in Pollett [12] for that papers variant on LIOpen can be used to show this papers LIOpen proves:*

1. $(\exists w \leq_l \langle s,t\rangle)[(w)_1 \leq_l s \wedge (w)_2 \leq_l t \wedge A((w)_1,(w)_2)] \iff (\exists x \leq_l s)(\exists y \leq_l t)\ A(x,y)$.

2. $(\forall w \leq_l \langle s,t\rangle)[(w)_1 \leq_l s \wedge (w)_2 \leq_l t \supset A((w)_1,(w)_2)] \iff (\forall x \leq_l s)(\forall y \leq_l t)A(x,y)$.

*From our definitions LIOpen* $\subseteq (T\Sigma_k^{\log})'$, *so* $(T\Sigma_k^{\log})'$ *can prove this as well.*

**Lemma 3** $T\Sigma_1^{\log} = TAC^0$.

*Proof.* Let $\Psi \supseteq \Sigma_1^{\log}$ be the class of formulas for which $T\Sigma_1^{\log}$ proves comprehension. $\Psi$ will be closed under term substitution. It suffices to show $\Psi$ is closed under $\neg$, $\wedge$, and $(\exists x \leq_l |t|)$. Let $A, B \in \Psi$. For closure under negation, notice that given $COMP_A(t)$ and $COMP_{\neg \mathrm{BIT}(i,w)=1}(t)$ it is straightforward to prove $COMP_{\neg A}(t)$. Similarly, from $COMP_A(t)$, $COMP_B(t)$, and $COMP_{\mathrm{BIT}(i,w)=1 \wedge \mathrm{BIT}(i,v)=1}(t)$ it is not hard to prove $COMP_{A\wedge B}(t)$. Lastly, consider $D := (\exists j \leq_l |z|)A(i,j,z,x)$. Let $C := A(\beta_t(j,i),j,z,x)$. Then $COMP_D$ follows from $COMP_C$ and $COMP_{(\exists j \leq_l |z|)(\mathrm{BIT}(i,\beta_t(j,v))=_l 1)}$. $\Box$

Note this lemma does not imply $(T\Sigma_k^{\log})' = TAC^0$ or even $(TAC^0)' := \cup_k (T\Sigma_k^{\log})'$ equals $TAC^0$. However, it does show that our restriction on cut will be important, as it restricts our ability to compose functions.

## 3 Definability

Let $\Psi$ be a set of formulas. A deduction system $T$ can $\Psi$-*define* a function $f(x)$, if there is a formula $A_f(x,y) \in \Psi$ such that $T \vdash \forall x \exists! y\ A_f(x,y)$ and $\mathbb{N} \models A_f(x,y) \Leftrightarrow f(x) = y$.

If, in addition, $T$ proves $y \leq_l t$ — that is, if $T \vdash \forall x \exists! y \leq_l t\ A_f(x,y)$ — then we say $T$ can *boundedly* $\Psi$-*define* $f$. A predicate is $\Delta_i^b$ with respect to a theory $T$ if it is provably equivalent to both a $\Sigma_i^{\mathsf{b}}$-formula and a $\Pi_i^{\mathsf{b}}$-formula. It should be observed that in a system where Parikh's theorem holds, such as $TAC^0$ or $S_2^i$, the notions of bounded definability and usual definability provably coincide. However, in the deduction systems above that have a restriction on cut, Parikh's Theorem might not provably hold.

**Definition 6** *1.* $(\mu x \leq_l |z|)[\phi]$ *returns the least* $x \leq_l |z|$ *such that* $\phi$ *holds and returns* $|z|+1$ *if no such value exists.*

*2.* $(\# x \leq_l |z|)[\phi]$ *returns the number of* $x \leq_l |z|$ *such that* $\phi$ *holds.*

*3.* $f$ *is defined by* $|\tau|$-bounded primitive recursion ($BPR^{|\tau|}$) *from multi-functions* $g$ *and* $h$*, and terms* $t \in L$*, and* $r \in L$ *if there is an* $\ell \in \tau$ *and a function* $F$ *such that*

$$\begin{array}{rcl} F(0,\vec{x}) & = & g(\vec{x}) \\ F(n +_{\ell(t)} 1, \vec{x}) & = & \min(h(n,\vec{x},F(n,\vec{x})), r(n,\vec{x})) \\ f(n,\vec{x}) & = & F(|\ell(t(n,\vec{x}))|, \vec{x})\ . \end{array}$$

**Definition 7** *A function* $f$ *is* $\Psi$-*comprehension defined in* $T$ *if there is a* $\Psi$-*formula* $A$ *such that* $f$ *can be* $\Psi$-*defined in* $T$ *by proving* $(\forall x)\ COMP_A(t)$*. That is,* $\mathbb{N} \models BIT(i,f(x)) = 1 \iff A(x,i)$*,* $T \vdash \forall x \exists y \leq_l t \forall i \leq_l |t| BIT(y,i) = 1 \iff A(x,i)$*, and* $T$ *proves that* $y$ *is unique.*

Uniqueness of $y$ is usually proven by using a notion of bit-extensionality. which says that if two numbers have the same bit string then they are equal. This can be proven in *LIOpen* from Pollett [12]. Comprehension definition has some nice properties with respect to the theories we will be considering.

**Lemma 4** *1. Let* $k > 0$*. If* $s$ *and* $m$ *are* $L$*-terms and* $f$*,* $g$*, and* $h$ *are* $\mathsf{B}(\Sigma_k^{\log})$*-comprehension defined in* $(T\Sigma_k^{\log})'$*, then so are*

*(a)* $f(s)$*,* $\langle f, g\rangle$*,* $(f)_j$*, and* $\beta_s(i,w)$*;*

*(b)* $\mathrm{cond}(f,g,h)$*, provided* $f \leq_l 1$*;*

(c) $\sum_{j=0}^{|s|-1} f(j)2^{j\cdot 2^{||m||}}$, *where* $m := f^+(|s|, x)$.

2. *If* $\phi \in \mathsf{B}(\Sigma_k^{\log})$, *then* $(T\Sigma_{k+1}^{\log})'$ *can* $\Sigma_{k+1}^{\log}$*-comprehension define* $(\mu x \leq_l |z|)[\phi]$.

*Proof.* That $f(s)$ is $\mathsf{B}(\Sigma_k^{\log})$-comprehension definable follows from term substitution of $s$ for the parameter of $f$ in its defining formula. For the rest of (1) we only show how to do $\mathrm{cond}(f, g, h)$ and $\sum_{i=0}^{|t|-1} f(i)2^{i\cdot 2^{||m||}}$ as the rest can be done similarly. Let $f$,$g$, and $h$ be $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined using formulas $A_f$, $A_g$ and $A_h$. We can $\mathsf{B}(\Sigma_k^{\log})$-comprehension define $\mathrm{cond}(f, g, h)$ using the formula

$$(A_f(0, x) \wedge A_g(i, x)) \vee (\neg A_f(0, x) \wedge A_h(i, x))\ .$$

Notice we are using that $f \leq_l 1$ in this case, so only its 0th bit matters.

To define $\sum_{j=0}^{|s|-1} f(j, x)2^{j\cdot 2^{||m||}}$ we can use the formula

$$A_f(i \dot{-}_s \mathrm{PAD}(\mathrm{MSP}(i, ||m||), |m|), \mathrm{MSP}(i, ||m||), x)$$

For (2), given $\phi$ we can $\Sigma_{k+1}^{\log}$-comprehension define $(\mu x \leq_l |z|)[\phi]$ using

$$\begin{aligned} A[i] \quad := \quad & (\exists j \leq_l |z|)[\mathrm{BIT}(i, j) =_l 1 \wedge \phi(j) \wedge \\ & (\forall j' \leq_l |z|)(j' \leq_z (j +_z 1) \supset \neg\phi(j'))] \vee \\ & (\forall j \leq_l |z|)[\neg\phi(j) \wedge \mathrm{BIT}(i, |z| + 1) = 1]. \end{aligned}$$

Inside the $[\cdots]$ is a $\mathsf{B}(\Sigma_k^{\log})$-formula, so prenexifying shows the result. □

**Definition 8** *Given* $t \in L$ *we define a monotonic term* $t^+$ *called the* dominator *for* $t$ *by induction on the complexity of* $t$.

- *If* $t$ *is a constant or a variable, then* $t^+ := t$.
- *If* $t$ *is* $\mathrm{LSP}(f, g)$ *or* $\mathrm{MSP}(f, g)$, *then* $t^+ := f^+$.
- *If* $t$ *is* $f \circ g$ *for any binary operation* $\circ$ *other than* $LSP$ *or* $MSP$, *then* $t^+ := f^+ \circ g^+$.
- *If* $t$ *is* $|f|$ *then* $t^+ := |f^+|$.

**Lemma 5** *(1)* $TAC^0$ *proves its* $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$*-definable functions are closed under composition. (2)* $T := TAC^0 + \Sigma_1^{\mathsf{b}}(\mathsf{LH})$*-*$LIND^{\tau}$ *proves its* $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$*-definable functions are closed under* $BPR^{|\tau|}$. *(3) If* $i > 0$, *then* $S_2^i$ *in the union of the languages of Buss [3] and of this paper is conservative over both the* $S_2^i$ *of Buss [3] and the* $S_2^i$ *of this paper.*

*Proof.* For (1), suppose $f = h(g_1(\vec{x}_1), \ldots g_n(\vec{x}_n))$ and that $TAC^0$ can $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-define $h(z_1, \ldots z_n)$ and $g_j(\vec{x}_j)$ where $1 \leq j \leq n$. Then there are $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-formulas $H$, $G_1, \ldots, G_n$ such that $TAC^0 \vdash (\forall \vec{z})(\exists y \leq_l t)\ H(\vec{z}, y)$ and $TAC^0 \vdash (\forall \vec{x}_j)(\exists y \leq_l t_j)\ G_j(\vec{x}_j, y)$, for $1 \leq j \leq n$.

$$TAC^0 \vdash (\forall \vec{x}_1) \cdots (\forall \vec{x}_n)(\exists y \leq_l t)(\exists y_1 \leq_l t_1) \cdots (\exists y_n \leq_l t_n)\ (G_1(\vec{x}_1, y_1) \\ \wedge \cdots \wedge G_n(\vec{x}_1, y_1) \wedge H(y_1, \ldots y_n, y)).$$

Since $LIOpen \subseteq TAC^0$ can do pairing, the formula inside the $(\exists y \leq t)$ is equivalent to a $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-formula in $TAC^0$.

For (2), suppose $f$ is obtained by $BPR^{|\tau|}$ from $g$ and $h$ which are $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-definable in $T$ where $r, t \in L$, and $\ell \in \tau$. Let $G$ and $H$ be the $\Sigma_i^{\mathsf{b}}(\mathsf{LH})$-graphs of $g$ and $h$ such that $T \vdash (\forall \vec{x})(\exists y \leq t_1)\ G(\vec{x}, y)$ and $T \vdash (\forall n, \vec{x}, u)(\exists v \leq t_2)\ H(n, \vec{x}, u, v)$. We can assume that $r(0, \vec{x}) \leq t_1(\vec{x})$. So let $A(n, \vec{x}, w, y)$ be

$$\begin{aligned} & G(\vec{x}, \beta_{r(0,\vec{x})}(0, w))) \wedge \\ & \quad \beta_{r^+(|\ell(t)|)}(n, w) = y \wedge \\ & \quad (\forall j < |\ell(t)|)((H(j, \vec{x}, \beta_{r^+(|\ell(t)|,\vec{x})}(j, w), \beta_{r^+(|\ell(t)|,\vec{x})}(Sj, w)) \\ & \quad \wedge \beta_{r^+(|\ell(t)|,\vec{x})}(Sj, w) < r(n, \vec{x})) \vee \beta_{r^+(|\ell(t)|,\vec{x})}(Sj, w) = r(n, \vec{x})) \end{aligned}$$

and let $B(n, \vec{x})$ be $(\exists y \leq r)(\exists w \leq 2 \cdot (|\ell(t)| \#_l r^+))\ A(n, \vec{x}, z, w, y)$. Let $F(n, \vec{x}, y)$ denote the formula within the $(\exists y \leq r)$. This formula is equivalent to a $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-formula in $T$ and we can define $f$ if we can show

$$(\forall \vec{x}, n)(\exists y \leq r)\ F(\ell(t(n, \vec{x})), \vec{x}, y).$$

So it suffices to show $(\forall \vec{x}, n)\ B(|\ell(t)|, \vec{x})$. Now $B$ is also equivalent to a $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-formula, so $T$ can use $LIND_B^\tau$ axioms. Since $T$ proves $(\forall \vec{x})(\exists y \leq t_1)G$, it proves $B(0, \vec{x})$. Suppose $T \vdash B(m, \vec{x})$, where $m \leq |\ell(t)|$. So there are $v, w, y$ satisfying $A(m, \vec{x}, w, y)$. If we set $y' = h(m, \vec{x}, y)$, and

$$w' = y' \cdot 2^{\min((m+1) \cdot 2^{||r^+||}, |\ell(c)| \cdot 2^{||r^+||})} + \mathrm{LSP}(w, (m+1) \cdot 2^{||r^+||})\ ,$$

then $T \vdash A(m+1, \vec{x}, z, w', y')$. Here we use $(m+1) \cdot 2^{||r^+||}$ to abbreviate $\mathrm{CAT}(m, 2^{||r^+||})$. Hence, $T \vdash B(m+1, \vec{x})$. By the $LIND_B^\tau$ axioms, $T \vdash (\forall \vec{x}, n) B(|\ell(t)|, \vec{x})$.

This same argument, as well as the arguments of Lemma 4, shows that the $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-definable functions of $S_2^1$ are closed under composition and $BPR^{\{|id|\}}$. Using $BPR^{\{|id|\}}$ it is then relatively easy to $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-define $S$,+, $\cdot$, and $2^{|x||y|}$ in our version of $S_2^1$. Using $LIND$ we can prove the various definitional properties of these functions. Similarly, in Buss' version of $S_2^1$

one can define PAD, MSP, LSP, and $\#_l$. Using these definitions one can in a straightforward but tedious fashion show that $S^i_2$ in the union of these two languages is conservative over $S^i_2$ in either language. □

**Definition 9** *We denote* $\cup_k \Sigma^{\log}_k$ *by* $\mathsf{LH}$. *We denote the closure of the* $\mathsf{LH}$-*functions under* $BPR^{|\tau|}$ *by* $\mathsf{LH}_{|\tau|}$.

**Theorem 3.1** *Let* $k > 1$. *Then:*

1. $(T\Sigma^{\log}_k)'$ *can* $\mathsf{B}(\Sigma^{\log}_k)$-*comprehension define the* $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$ *functions.*

2. *Both* $TAC^0$ *and* $(TAC^0)'$ *can* $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-*define the functions in* $\mathsf{LH}$ = *uniform* $\mathsf{AC}^0$.

3. $TAC^0$+$\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-$LIND^\tau$ *can* $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-*define the functions in* $\mathsf{LH}_{|\tau|}$.

*Proof.* (1) follows from Lemma 2 and the definition of $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$ function as being a poly-bounded function whose bit-graph is in $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$. That $(T\Sigma^{\log}_k)'$ can prove the value produced by using $\mathsf{B}(\Sigma^{\log}_k)$-*COMP* is unique follows immediately from our definition of equality applied to the bit-string produced by a $\mathsf{B}(\Sigma^{\log}_k)$-*COMP* axiom. (2) follows from (1) since $TAC^0 := \cup_k T\Sigma^{\log}_k$ and that the $\mathsf{LH}$ functions are just the $\cup_k\mathsf{B}(\Sigma_k - \mathsf{LOGTIME})$ functions. (3) follows from (2) and Lemma 5. □

We next briefly describe how to establish the converse of the above result. First we define a witness bounding term and witness predicate for $\mathsf{LE}\Pi^{\log}_{k+1}$-formulas as follows:

- If $A(\vec{a}) \in \mathsf{LU}_{\{|id|\}}\mathsf{B}(\Sigma^{\log}_k)$ then $t_A = 0$ and $WIT^k_A(w, \vec{a}) := A(\vec{a}) \wedge w = 0$.

- If $A(\vec{a})$ is of the form $(\exists x \leq_l t)\mathsf{B}(x, \vec{a})$ where $\mathsf{B}(x, \vec{a}) \in \mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma^{\log}_k)$ then $t_A := t$ and

$$WIT^k_A(w, \vec{a}) := w \leq_l t \wedge \mathsf{B}(w, \vec{a}) \ .$$

The following lemma follows from the definition of witness predicate:

**Lemma 6** *Let* $A(\vec{a}) \in \mathsf{LE}\Pi^{\log}_{k+1}$. *Then:*

$WIT^k_A$ *is a* $\mathsf{LU}_{\{|id|\}}\mathsf{B}(\Sigma^{\log}_k)$ *-predicate.*

*LIOpen* $\vdash (\exists w \leq_l t_A(\vec{a}))\, WIT^k_A(w, \vec{a}) \supset A(\vec{a})$.

We extend the witness predicate to cedents in the standard way given in Buss [3]. The next theorem is used to prove the converse of Theorem 3.1.

**Theorem 3.2** *Suppose $k \geq 1$ and*

$$(T\Sigma_k^{\log})' \vdash \Gamma \to \Delta \ ,$$

*where $\Gamma$ and $\Delta$ are cedents of $\mathsf{LE}\Pi_{k+1}^{\log}$-formulas. Let $\vec{a}$ be the free variables in this sequent. Then there is a function $f$ which is $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined in $(T\Sigma_k^{\log})'$ by $A_f$ such that $(T\Sigma_k^{\log})'$ proves:*

$$(\forall i \leq |t|)(\mathrm{BIT}(i,y) = 1 \Leftrightarrow A_f(i,w,\vec{a})),\ WIT^k_{\wedge\Gamma}(w,\vec{a}) \to WIT^k_{\vee\Delta}(y,\vec{a}). \quad (1)$$

Notice that the sequent in (1) contains only $\mathsf{LU}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-formulas, so $(T\Sigma_k^{\log})'$ can reason about this sequent using cuts. From now on we abbreviate a sequent like (1) as

$$(T\Sigma_k^{\log})' \vdash WIT^k_{\wedge\Gamma}(w,\vec{a}) \to WIT^k_{\vee\Delta}(f(w,\vec{a}),\vec{a}).$$

*Proof.* The proof is by induction on the number of sequents in a $(T\Sigma_k^{\log})'$ proof of $\Gamma \to \Delta$. By cut elimination, we can assume that all the sequents in the proof contain only $\mathsf{LEU}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$- formulas. In the case of a proof involving only an initial sequent, the only kind of initial sequents involving $\mathsf{EU}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-formulas are $\mathsf{B}(\Sigma_k^{\log})$-$COMP$ axioms. So in all other cases we can create a witness function for the succedent using pairings of the 0 function. (Notice that this is true even of $\mathsf{B}(\Sigma_k^{\log})$-$LIND$ axioms.) Let $A(i,\vec{a})$ be a $\mathsf{B}(\Sigma_k^{\log})$-formula. Then a witness for a $COMP_A(t)$ axiom will be just the $\mathsf{B}(\Sigma_k^{\log})$-function given by the bit-graph of $A$. The inductive step now breaks into cases according to the last inference in the $(T\Sigma_k^{\log})'$ proof. Most of the cases are similar to those in previous witnessing arguments so we only show three cases: the ($\forall$ : right) case, the *cut*-case, and the ($AND$ : right) case.

**($\wedge$:right case)** Suppose we have the inference:

$$\frac{\Gamma \to A, \Delta \qquad \Gamma \to B, \Delta}{\Gamma \to A \wedge B, \Delta}$$

The induction hypothesis gives $g$ and $h$ that are $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined such that

$$\begin{array}{lcl}(T\Sigma_k^{\log})' & \vdash & WIT^k_{\wedge\Gamma}(w,\vec{a}) \supset WIT^k_{A\vee\Delta}(g(w,\vec{a}),\vec{a}) \\ (T\Sigma_k^{\log})' & \vdash & WIT^k_{\wedge\Gamma}(w,\vec{a}) \supset WIT^k_{B\vee\Delta}(h(w,\vec{a}),\vec{a}).\end{array}$$

Notice that since we only can have conjunctions of open formulas in our proof, $A$ and $B$ must be open, which means that $WIT^k_A$ is an *open*-formula in $BASIC$. So we can express $A$ as an $L$-term $f_W$ using $\mathrm{K}_{\leq_l}$, $\mathrm{K}_\wedge$, and $\mathrm{K}_\neg$. We define the term $k$ as

$$k(v, w, \vec{a}) := \mathrm{cond}(f_W(v, \vec{a}), v, w)\ .$$

Now define $f$ by

$$f(w, \vec{a}) := \langle 0, k((g(w, \vec{a}))_1, (h(w, \vec{a}))_2, \vec{a}\rangle.$$

The function $f$ is $\mathsf{B}(\Sigma^{\log}_k)$-definable by Lemma 4. Since the formula $A \wedge B$ must be open, we have $WIT^k_{A\wedge B} = A \wedge B \wedge w = 0$ and which takes 0 as a witness. Thus the function $f$ provides a witness, if needed, to the remaining formulas in the succedent and one can verify that

$$(T\Sigma^{\log}_k)' \vdash WIT^k_{\wedge\Gamma}(w, \vec{a}) \to WIT^k_{A\wedge B\vee\Delta}(f(w, \vec{a}), \vec{a}).$$

**(Cut rule case)** Suppose we have the inference:

$$\frac{\Gamma \to A, \Delta \qquad A, \Gamma \to \Delta}{\Gamma \to \Delta}$$

The induction hypothesis gives $g$ and $h$ that are $\mathsf{B}(\Sigma^{\log}_k)$-comprehension defined such that

$$\begin{aligned}(T\Sigma^{\log}_k)' &\vdash WIT^k_{\wedge\Gamma}(w, \vec{a}) \to WIT^k_{A\vee\Delta}(g(w, \vec{a}), \vec{a})\\ (T\Sigma^{\log}_k)' &\vdash WIT^k_{A\wedge\Gamma}(w, \vec{a}) \to WIT^k_{\vee\Delta}(h(w, \vec{a}), \vec{a}).\end{aligned}$$

We define the function $k$ as

$$k(v, w, \vec{a}) := \mathrm{cond}(f_W(v, \vec{a}), v, w)$$

Here $f_W$ is as in the $\wedge$:right case. We define the function $f$ to be

$$f(w, \vec{a}) := k((g(w, \vec{a}))_1, h(\langle 0, w\rangle, \vec{a}))\ .$$

Since we are considering $(T\Sigma^{\log}_k)'$-proofs, $A$ must be an $\mathsf{LU}_{\{|id|\}}\mathsf{B}(\Sigma^{\log}_k)$-formula. By Lemma 4, $f$ is in $\mathsf{B}(\Sigma^{\log}_k)$-comprehension definable and it is easy to see that

$$(T\Sigma^{\log}_k)' \vdash WIT^k_{\wedge\Gamma}(w, \vec{a}) \to WIT^k_{\vee\Delta}(f(w, \vec{a}), \vec{a}).$$

**(∀:right case)** Suppose we have the inference:

$$\frac{b \leq_l t, \Gamma \to A(b), \Delta}{\Gamma \to (\forall x \leq_l t)A(x), \Delta}$$

By the induction hypothesis there is a $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined function $g$ such that

$$(T\Sigma_k^{\log})' \vdash WIT^k_{b\leq_l t\wedge\Gamma}(w, \vec{a}, b) \to WIT^k_{A\vee\Delta}(g(w, \vec{a}, b), \vec{a}, b) \ .$$

By cut-elimination, $(\forall x \leq_l t)A(x)$ is a $\mathsf{LE}\Pi^{\log}_{k+1}$-formula, so $t$ must be of the form $t = |s|$ and $A \in L\Sigma_k^{\log}$. Let $y$ be $(\mu i \leq_l |s|)\neg A(i)$. Using Lemma 4, we $\mathsf{B}(\Sigma_k^{\log})$-comprehension define a function $h$ that outputs

$$\sum_{i=0}^{|s|} g(\langle 0, w\rangle, \vec{a}, i)2^{\min(i\cdot 2^{||m||})} \ ,$$

where $m$ is as in the lemma. The first component of the ordered pair is 0 because $WIT_{b\leq_l t}(w, b) = b \leq_l t \wedge w = 0$. Since each of the definitions of sums and projections in Lemma 4, involved only simple substitutions into the original formula being comprehended over, $(T\Sigma_k^{\log})'$ can prove simple facts about $h$ provided $h$ is suitably defined. In particular, it can show that $\beta_{2^{||m||}}(i, h) = g(\langle 0, w\rangle, \vec{a}, i)$. Let $f$ be $\beta_{2^{||m||}}(y, h)$. This is in $(T\Sigma_k^{\log})'$ by Lemma 4 and

$$(T\Sigma_k^{\log})' \vdash WIT^k_{\Gamma}(w, \vec{a}) \to WIT^k_{\forall x\leq_l|s|\ A\wedge\Delta}(f(w, \vec{a}), \vec{a}) \ .$$

This completes the cases and the proof. □

**Corollary 2** *For all $k \geq 1$, $(T\Sigma_k^{\log})'$ proves its boundedly $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-definable functions can be $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined. Hence, for any $k > 1$, the boundedly $\mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$-definable functions of $(T\Sigma_k^{\log})'$ are precisely $\mathsf{B}(\Sigma_k\text{-}\mathsf{LOGTIME})$.*

*Proof.* Suppose $(T\Sigma_k^{\log})'$ proves $\to (\exists y \leq_l t)A(x, y)$ and $A \in \mathsf{U}_{\{|id|\}}\mathsf{B}(\Sigma_k^{\log})$. A witness for the empty cedent is 0. So Theorem 3.2 gives the desired $\mathsf{B}(\Sigma_k^{\log})$-comprehension defined function $f$ so that $(T\Sigma_k^{\log})' \vdash A(x, f(0, x))$. □

**Theorem 3.3** *(1) The $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-definable functions of $TAC^0$ are precisely $\mathsf{LH}$. (2) The $\Sigma_1^{\mathsf{b}}(\mathsf{LH})$-defined functions of $TAC^0 + \Sigma_1^{\mathsf{b}}(\mathsf{LH})$-$LIND^{\tau}$ are precisely $\mathsf{LH}_{|\tau|}$.*

*Proof.* That these theories can define their respective function classes is the content of Theorem 3.1.

For the other direction, first that note that because Parikh's Theorem does hold in these theories (one can use the argument in Hajek Pudlak [4] to show this) if one of these theories can prove $\exists y A(x,y)$, then there is a term $t \in L$ such that it can prove $(\exists y \leq t)A(x,y)$. So we can prove the converse to Theorem 3.1 by carrying out a witnessing argument in much the same way as was done in Theorem 3.2. We first modify the witness predicate as follows:

- If $A(\vec{a}) \in L \cup_k \Sigma^{\log}_k$ then $t_A := 0$ and $WIT_A(w,\vec{a}) := A(\vec{a}) \wedge w = 0$.
- If $A(\vec{a})$ is of the form $(\exists x \leq_l t)B(x,\vec{a})$ where $B(x,\vec{a}) \in \cup_k \Sigma^{\log}_k$ then $t_A := t$ and
$$WIT_A(w,\vec{a}) := w \leq_l t \wedge B(w,\vec{a}) \ .$$
- If $A(\vec{a})$ is of the form $(\exists x \leq_l t)(\exists y \leq_l s)B(x,y,\vec{a})$ where $B(x,\vec{a}) \in \cup_k \Sigma^{\log}_k$ then $t_A := \langle t, s\rangle$ and
$$WIT_A(w,\vec{a}) := w \leq_l t_A \wedge B((w)_1,(w)_2,\vec{a}) \ .$$

We extend $WIT$ to cedents in the same way as in Buss [3]. We will give a proof of the the theorem only for $T := TAC^0 + \Sigma^{\mathsf{b}}_1\text{-}LIND^{\tau}$, the other results are similar. We argue by induction on the number of sequents in a $T$ proof of $\Gamma \to \Delta$ that there is an $\mathsf{LH}_{|\tau|}$ function $f$ such that $T \vdash WIT_{\wedge\Gamma}(w,\vec{a}) \supset WIT_{\vee\Delta}(f(w,\vec{a}),\vec{a})$. Most of the cases are handled in the same way as in the argument of Theorem 3.2. The cut-case changes and we need to handle $\Sigma^{\mathsf{b}}_1\text{-}LIND^{\tau}$.

**(Cut rule case)** Suppose we have the inference:

$$\frac{\Gamma \to A, \Delta \qquad A, \Gamma \to \Delta}{\Gamma \to \Delta} \ .$$

By the induction hypothesis there are $\mathsf{LH}$-functions $g$ and $h$ such that

$$\begin{array}{lcl} T & \vdash & WIT_{\wedge\Gamma}(w,\vec{a}) \supset WIT_{A\vee\Delta}(g(w,\vec{a}),\vec{a}) \\ T & \vdash & WIT_{A\wedge\Gamma}(w,\vec{a}) \supset WIT_{\vee\Delta}(h(w,\vec{a}),\vec{a}). \end{array}$$

We define the function $k$ by

$$k(v,w,\vec{a}) := \text{cond}(f_W(v,\vec{a}),v,w)$$

Here $f_W$ is as in the $\wedge$:right case of Theorem 3.2. We define the function $f$ to be

$$f(w,\vec{a}) := k((g(w,\vec{a}))_1, h(\langle (g(w,\vec{a}))_1, w\rangle, \vec{a}))\ .$$

By Lemma 5, $f$ is $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-definable, and since $\mathsf{LH}_{|\tau|}$ is closed under composition, $f$ will be in $\mathsf{LH}_{|\tau|}$. It is not hard to show that

$$TAC^0 \vdash WIT_{\wedge\Gamma}(w,\vec{a}) \supset WIT_{\vee\Delta}(f(w,\vec{a}),\vec{a})\ .$$

**($\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-$LIND^\tau$ case)** Without loss of generality (see Buss [3]) we can reformulate $\Sigma^{\mathsf{b}}_1$-$LIND^\tau$ as the following kind of induction inference:

$$\frac{A(b), \Gamma \rightarrow A(b +_{\ell(t)} 1), \Delta}{A(0), \Gamma \rightarrow A(|\ell(t)|), \Delta}$$

where $\ell \in \tau$. By hypothesis there is a $g \in \mathsf{LH}_{|\tau|}$ such that

$$T \vdash WIT_{A(b)\wedge\Gamma}(w,\vec{a}) \supset WIT_{A(b+_{\ell(t)}1)\vee\Delta}(g(w,\vec{a}),\vec{a}).$$

Let $h(m,w,\vec{a},b)$ be

$$\mathrm{cond}(WIT_{A(b+_{\ell(t)}1)\vee\Delta}(m,\vec{a},b), m, g(\langle m, \beta(2,w)\rangle, \vec{a}, b))$$

Define $f$ by $BPR^{|\tau|}$ in the following way

$$\begin{array}{rcl} F(0,w,\vec{a}) & = & \langle (w)_1, 0\rangle \\ F(b+1,w,\vec{a}) & = & \min(h(F(b,w,\vec{a}),w,\vec{a},b), t_{\vee A(Sb)\vee\Delta}) \\ f(u,w,\vec{a}) & := & h(\min(u,|\ell(t)|),w,\vec{a})\ . \end{array}$$

Recall that $t_{\vee A(b+_{\ell(t)}1)\vee\Delta}$ is a term guaranteed to bound a witness for $A(b +_{\ell(t)} 1) \vee \Delta$ (defined by using pairings of the terms bounding the witnesses for the individual formulas). It is easy to see that

$$T \vdash WIT_{A(0)\wedge\Gamma}(w,\vec{a}) \supset WIT_{A(0)\vee\Delta}(f(0,w,\vec{a}),\vec{a})$$

From this one can show that

$$\begin{array}{l} T \vdash WIT_{A(0)\wedge\Gamma}(w,\vec{a}) \wedge WIT_{A(b)\vee\Delta}(f(b,w,\vec{a}),b,\vec{a}) \\ \quad \supset WIT_{A(b+_{\ell(t)}1)\vee\Delta}(f(b +_{\ell(t)} 1, w, \vec{a}), b +_{\ell(t)} 1, \vec{a})\ , \end{array}$$

from which it follows by $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-$LIND^\tau$ that

$$T \vdash WIT_{A(0)\wedge\Gamma}(w,\vec{a}) \supset WIT_{A(|\ell(t)|)\vee\Delta}(f(|\ell(t)|,w,\vec{a}),\vec{a})\ .$$

This completes the cases we will show of the witnessing argument.

From this it follows that if $T$ $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-defines a function by proving $(\exists y \leq_l t)A(x, y)$, where $A$ is $(\exists z \leq_l t)B(x, y, z)$ and $B \in \cup_k(\Sigma^{\log}_k)$, then we get an $\mathsf{LH}$ function $f$ such that $WIT_A(x, (f(0, x))_1, (f(0, x))_2)$. This implies $A(x, (f(0, x))_1)$, so the function defined by $T$ was in $\mathsf{LH}$. $\Box$

The next two results are easy modifications of results in Pollett [12]. Since $S^i_2$ has $\Sigma^{\mathsf{b}}_1$-quantifier replacement [3, 12], it can actually show that any $\Sigma^{\mathsf{b}}_i(\mathsf{LH})$-formula (resp. $\Pi^{\mathsf{b}}_i(\mathsf{LH})$-formula) is equivalent to a $\Sigma^{\mathsf{b}}_i$-formula (resp. $\Pi^{\mathsf{b}}_i$-formula).

**Theorem 3.4** $(i \geq 1)$ *Suppose that for all $\ell \in \tau$, $\ell \in O(\{|\dot{x}|\})$. Let $2^{\dot{\tau}}$ be the set of terms $2^\ell$ where $\ell \in \dot{\tau}$. Then*

1. $TAC^0 + \Sigma^{\mathsf{b}}_i(\mathsf{LH})\text{-}LIND^{2^{\dot{\tau}}} \preceq_{\mathsf{B}(\Sigma^{\mathsf{b}}_{i+1})} TAC^0 + \Sigma^{\mathsf{b}}_{i+1}(\mathsf{LH})\text{-}LIND^{\tau}$;

2. *the $\Delta^b_{i+1}(\mathsf{LH})$-predicates of both these classes are $\mathsf{P}^{\Sigma^{\mathsf{p}}_{i-1}}(|\dot{\tau}|)$.*

**Theorem 3.5** $(i \geq 0, k \geq 2)$ *The $\Delta^b_{i+k}(\mathsf{LH})$-predicates of $S^i_2$ are precisely $\mathsf{P}^{\Sigma^{\mathsf{p}}_{i+k-1}}(1)$.*

# 4 Separations in $(T\Sigma^{\log}_k)'$ and $TAC^0$

We begin by showing that there is a simple universal predicate for $\check{\Sigma}^\tau_k$.

**Lemma 1** *Let $\tau$ contain only terms which are $\Omega(\log n)$. Then there is an $L$-formula $U_k(e, x, z)$ such that for any $\phi(x) \in \check{\Sigma}^\tau_k$ there exist a fixed number $e_\phi$, a term $\ell \in \tau$, and a term $t \in L$ such that*

$$LIOpen \vdash U_k(e_\phi, x, \ell(t(x))) \equiv \phi(x)$$

*and such that for all terms $\ell \in \tau$, we have $U_k(e, x, \ell(t(x))) \in \Sigma^\tau_k$.*

*Proof.* First note that since we are assuming terms in $\tau$ are nondecreasing, we can replace the innermost quantifier bounds in $\phi$ with bounds that are a function (involving MSP) of $|\ell(\#^m_l(x))|$ and replace all other quantifier bounds in $\phi$ with bounds of the form $\ell(\#^m_l(x))$ for some large enough $m$. We can push the actual bounds into the open-matrix. This works even for the innermost quantifier because of the part of our definition of $\Sigma^\tau_0$ and $\Pi^\tau_0$ involving $h$-boundedness with respect to the innermost quantifying variable.

So using $K_{\leq_l}$, $K_\neg$, and $K_\wedge$ to rewrite the open matrix as a single equation, $\phi$ can be written provably in *BASIC* in the form:

$$(\exists y_1 \leq \ell(\#_l^m(x))) \cdots (Qy_k \leq \ell(\#_l^m(x)))[(Q'y_{k+1} \leq ms(x))\ (t_1 \leq_l 0) \wedge \\ (Q'y_{k+2} \leq ms(x))(Q'y_{k+3} \leq \mathrm{MSP}(||\ell(\#_l^m(x))||, |\ell(\#_l^m(x))|_4))(t_2 \leq_l 0)],$$

where $ms(x) := \mathrm{MSP}(|\ell(\#_l^m(x))|, |||\ell(\#_l^m(x))|||)$ and the quantifiers $Q$ and $Q'$ depend on whether $i$ is even or odd. Here all variables in $t$ are $|\ell|$-bounded. We fix some coding scheme for the 9 symbols of $L$. We use $\ulcorner\urcorner$ to denote the code for some symbol, e.g., $\ulcorner \leq_l \urcorner$ is the code for $\leq_l$. We also have codes for $\ulcorner\beta^{y_1}\urcorner, \ldots, \ulcorner\beta^{y}_{k+3}\urcorner, \ulcorner\beta^x\urcorner$, and codes $\ulcorner|\ell(y_1)|\urcorner, \ldots, \ulcorner|\ell(y_{k+3})|\urcorner, \ulcorner|\ell(x)|\urcorner$. We choose our coding so that all codes require less than $|2k+14|$ bits and we use 0 as $\ulcorner\mathrm{NOP}\urcorner$ meaning no operation. The code for a term $t$ is a sequence of blocks of length $|2k+14|$ that write out $t$ in postfix order. So $\mathrm{PAD}(|\ell(x)|, |\ell(y_1)|)$ would be coded as the three blocks $\ulcorner|\ell(x)|\urcorner\ulcorner|\ell(y_1)|\urcorner\ulcorner\mathrm{PAD}\urcorner$. The symbol $\ulcorner\beta^x\urcorner$ is codes the function which takes two arguments $a$ and $b$ and returns $\beta_a(b,x)$. The symbols $\ulcorner\beta^{y_i}\urcorner$ have a similar function. The code for a $\Sigma_k^\tau$-formula will be the pair of the codes of the innermost two terms.

We now describe $U_k(e,x,z)$. It will be obtained from the formula

$$(\exists w \leq z)(\exists w' \leq z)(\exists y_1 \leq z)(\forall j \leq |e|)(\forall y_2 \leq z) \cdots (Qy_k \leq z) \\ ((Q'y_{k+1} \leq \mathrm{MSP}(|z|, ||z||))\ \phi_k((e)_1, j, x, \vec{y}, w) \wedge \\ (Q'y_{k+2} \leq \mathrm{MSP}(|z|, ||z||))(Q'y_{k+3} \leq \mathrm{MSP}(||z||, |z|_3))\phi_{k'}((e)_1, j, x, \vec{y}, w')$$

after grouping together like quantifiers. Here $\phi_k$ consists of a statement saying $w$ codes a postfix computation of the term given by $(e)_1$ and $\phi_{k'}$ consists of a statement saying $w'$ codes a postfix computation of the term given by $(e)_2$. For $\phi_k$ this amounts to checking conditions

$$[\beta_{2k+14}(j,e) = \ulcorner|\ell(x)|\urcorner \supset \beta_B(j,w) = |\ell(x)|]\ \wedge$$

$$[\beta_{2k+14}(j,e) = \ulcorner PAD\urcorner \supset \beta_B(j,w) = \mathrm{PAD}(\beta_B(j \dot{-} 2, w), \beta_B(j \dot{-} 1, w)]\ \wedge$$

$$\cdots$$

$$[\beta_{2k+14}(j,e) = \ulcorner NOP\urcorner \supset \beta_B(j,w) = \beta_B(j \dot{-} 1, w)].$$

Finally, $\phi_k$ has a condition saying $\beta_B(|e|, w) \leq_l 0$. Since any subterm of $t$ is no larger than a number which results from applying $\#_l$ to things that are $|\ell|$-bounded, the length of a $w$ that works can be bounded by $||\ell(x)||^k$ for some $k$. So $w$ is less than $2^{||\ell||^k}$. $B$ in the above can thus be of the from $2^{||\ell||^{k'}}$ for some $k' < k$ and this number will be less than $\ell$. which is less than $\ell(\#_l^m(x))$. Similar conditions are checked in the $\phi_{k'}$ case and the same argument shows $w'$ is less than $\ell$.

Since *LIOpen* can prove simple facts about projections from pairs, it can prove by induction on the complexity of the term $t$ (which is finite) in any $\Sigma_k^\tau$-formula $\phi(x)$ that $U_k(e_\phi, x, \ell(\#_l^m(x))) \equiv \phi(x)$. □

**Theorem 4.1** *LIOpen proves that* $\Sigma_k^{m\cdot\log} \neq \Pi_k^{m\cdot\log}$. *Hence, LIOpen proves that* mLH *is infinite.*

*Proof.* Let $A(x)$ be any $\Sigma_k^{m\cdot\log}$ formula. We show *LIOpen* $\vdash (\exists x A(x)) \not\equiv (\neg U_k(x, x, |\#_l^m(x)|))$. This suffices since $\neg U_k(x, x, |\#_l^m(x)|)$ is in $\Pi_k^{m\cdot\log}$. Let $e_A$ be the code $U_k$ for $A$. Then *LIOpen* proves $(\neg U_k^{\log}(e_A, e_A, |\#_l^m(x)|)) \equiv (\neg A(e_A))$ and, therefore, $(\neg U_k(e_A, e_A, |\#_l^m(x)|)) \not\equiv A(e_A)$.

Thus *LIOpen* proves for every $k$ that $\Sigma_k^{m\cdot\log} \neq \Pi_k^{m\cdot\log}$. Hence, *LIOpen* proves that mLH is infinite. □

**Theorem 4.2**

*$TAC^0$ proves* $\mathsf{LH} \neq \Sigma_k\text{-}\mathsf{TIME}(\log^{O(1)} n)$ *for any* $k > 0$.

*$TAC^0$ proves* $\mathsf{LH} \neq \Sigma_k^{\mathsf{b}}$ *for any* $k > 0$.

*$TAC^0$ proves* $\mathsf{LH} \neq \mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$ *for all* $k$ *and* $m$.

Notice that although $\mathsf{LH} \subseteq \Sigma_1^{\mathsf{b}}$, we are not claiming $TAC^0$ can prove this. Since $\mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$ is almost $\Sigma_k^{\log}$, except for a slightly larger outermost existential, (3) gives some evidence that $TAC^0$ proves LH is infinite. *Proof.*

The second statement can be proven in the same fashion as the first, so we will only prove the first and third statements. A universal predicate for $\Sigma_k\text{-}\mathsf{TIME}(\log^{O(1)} n)$ will be just $U_k(e, x, \#_l^e(|x|))$. The statement $\mathsf{LH} = \Sigma_k\text{-}\mathsf{TIME}(s)$ means that for any formula $A \in \cup_k \Sigma_k^{\log}$,

$$(\exists e_A)(\forall x) U_k(e_A, x, \#_l^e(|x|)) \equiv A\ ,$$

and that there is an LH formula $U$ such that

$$(\forall x) U(\langle e, x, \#_l^e(|x|)\rangle) \equiv U_k(e, x, \#_l^e(|x|))\ .$$

By De Morgans Laws, to prove the negation of $\mathsf{LH} = \Sigma_k\text{-}\mathsf{TIME}(\log^{O(1)} n)$ it suffices to prove the negation of any finite subset of this infinite family of statements. Let $\phi$ be the conjunction of the following statements:

$$(\exists e_{\neg U})(\forall x)\ U_k(e_{\neg U}, z, \#_l^{e_{\neg U}}(|x|) \equiv \neg U(z)$$

and

$$(\forall x)U\ (\langle e, x, \#_l^e(|x|)\rangle) \Leftrightarrow U_k(e, x, \#_l^e(|x|))$$

We will argue informally that $TAC^0 \vdash \phi \supset \neg\phi$ and so $TAC^0 \vdash \neg\phi$. First notice that $TAC^0$ proves $\phi \supset LIND_{U_k}$ and that $TAC^0$ proves $\phi$ implies any $\Pi_k$-$\mathsf{TIME}(\log^{O(1)} n)$ predicate can be written as a $\Sigma_k$-$\mathsf{TIME}(\log^{O(1)} n)$ formula. Using pairing, this means that $TAC^0$ can prove that any formula $A$ in the $\cup_m(\Sigma_m$-$\mathsf{TIME}(log^{O(1)}n))$ hierarchy is equivalent to a $\Sigma_k$-$\mathsf{TIME}(\log^{O(1)} n)$ predicate and so $TAC^0$ proves $LIND_A$. Suppose $U$ is a $\Sigma_m^{\log}$ predicate. Using $LIND$, $TAC^0$ can effectively reason about the standard diagonalization language $L$ (Mocas [10] Theorem 3.2.1) used to show $\Sigma_m^{\log} \neq \Sigma_m$-$\mathsf{TIME}(\log^{O(1)})$. This implies $\neg\phi$ since $\phi$ implies there is a $U_k$ code (and hence a $U$ code) $e_L$ for $\phi$, which says that $L \in \Sigma_k^{\log} \subseteq \Sigma_m^{\log}$.

The third statement is proven similarly. First one argues in $TAC^0$ that $\mathsf{LH} = \mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$ implies

$$\mathsf{LH} = \mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log} = \mathsf{U}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$$

since $\mathsf{LH}$ is closed under complement. This implies that

$$\mathsf{LH} = \cup_v \Sigma_v\text{-}\mathsf{TIME}(\log^{m+1} n \cdot \log n))\ .$$

$TAC^0$ can prove that the universal predicate $U$ for $\mathsf{E}_{|x|_{m+1}\#_l|x|}\Sigma_k^{\log}$ is equivalent to some $\Sigma_v^{\log}$-formula for some fixed $m$, and $TAC^0$ can diagonalize $\Sigma_v^{\log}$ from $\Sigma_v$-$\mathsf{TIME}(\log^{m+1} n \cdot \log n))$ as in the previous argument. $\Box$

# 5 Independence

In this section, we prove $TAC^0$ and $ZAC^0$ cannot prove that the polynomial time hierarchy collapses.

**Theorem 5.1** *If $ZAC^0 \subseteq S_2^i$ for any $i$, then the polynomial hierarchy collapses to $\mathsf{B}(\Sigma_{i+2}^{\mathsf{p}})$.*

*Proof.* $ZAC^0 \subseteq S_2^i$ implies $ZAC^0_{i+2} \subseteq S_2^i$. The $\Delta_{i+2}^b(\mathsf{LH})$-predicates of $S_2^i$ are $\mathsf{P}^{\Sigma_{i+1}^{\mathsf{p}}}(1)$ by Theorem 3.5. By Corollary 3.4, the $\Delta_{i+2}^b$-predicates of $ZAC^0_{i+2}$ are $\mathsf{P}^{\Sigma_{i+1}^{\mathsf{p}}}((\{|id|_{i+5}\}))$. It is not hard to exhibit complete problems for the latter class. Hence, if $ZAC^0_{i+2} \subseteq S_2^i$, then

$$\mathsf{P}^{\Sigma_{i+1}^{\mathsf{p}}}(1) = \mathsf{P}^{\Sigma_{i+1}^{\mathsf{p}}}((\{|id|_{i+5}\}))\ ,$$

and so for some $k$, $\mathsf{P}^{\Sigma^{\mathsf{p}}_{i+1}}[k] = \mathsf{P}^{\Sigma^{\mathsf{p}}_{i+1}}[k+1]$. The result then follows from Chang and Kadin [8, 7]. □

**Definition 10** *Define* $2 \uparrow 0(x) := x$, $2 \uparrow i+1(x) := 2^{2 \uparrow i(x)}$. *Let* $\tau_i$ *be the set of iterms of the form* $2 \uparrow j(p(|x|_j))$ *for* $j \geq i+3$ *and* $p$ *any polynomial. Let* $\tau_Z := \cup_i \tau_i$.

As a consequence of Theorem 3.4 and the fact that a statement provable in $ZAC^0$ must in fact be provable in $ZAC^0_i$ (recall $ZAC^0_{i+1}$ contains $ZAC^0_i$) for some large enough $i$, we have:

**Lemma 2** *1.* $(i > 0)$ $TAC^0 + \Sigma^{\mathsf{b}}_i(\mathsf{LH})\text{-}LIND^{\tau_i} \preceq_{\mathsf{B}(\Sigma^{\mathsf{b}}_{i+1})} ZAC^0$.

*2.* $(i > 0)$ *The* $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$*-definable functions of* $ZAC^0$ *are precisely* $\mathsf{AC}^0_{|\tau_Z|}$.

To prove that $ZAC^0$ cannot prove the collapse of the hierarchy, we first show that if $ZAC^0$ proves $\mathsf{PH}\downarrow$ then $ZAC^0 = S_2$. This is the content of the next theorem.

**Theorem 5.2** *If* $ZAC^0$ *proves that the polynomial hierarchy collapses then* $ZAC^0 = S_2$.

*Proof.* Since $ZAC^0 := \cup_i ZAC^0_i$, if $ZAC^0$ proves that the polynomial hierarchy collapses, then there must be an $i$ and a $k$ such that $ZAC^0_i$ proves that the $U_k$ of Lemma 1 is equivalent to a $\Pi^{\mathsf{b}}_k$-formula. Hence, $ZAC^0_i$ proves that $\Sigma^{\mathsf{b}}_k = \Pi^{\mathsf{b}}_k$. Since $ZAC^0_i \subseteq ZAC^0_{i+1}$, without loss of generality we can assume that $k \leq i$. It follows that $ZAC^0_i$ proves $\Sigma^{\mathsf{b}}_m\text{-}LIND^{\{|id|_{i+3}\}}$ for all $m$. So if we choose $m := 2i+9$ we get $TAC^0 + \Sigma^{\mathsf{b}}_m\text{-}LIND^{\{|id|_{i+3}\}} \subseteq ZAC^0_i$. Then $i+3$ applications of Theorem 3.4 show $S^i_2 \subseteq ZAC^0_i$. Since $ZAC^0_i$ proves $\Sigma^{\mathsf{b}}_k = \Pi^{\mathsf{b}}_k$ and $k < i$, it must contain $S^m_2$ for every $m$. □

**Theorem 5.3** *Parity is not* $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$*-definable in* $ZAC^0$*. Hence,* $ZAC^0$ *and* $TAC^0$ *cannot prove that the polynomial hierarchy collapses.*

*Proof.* By Lemma 2, the $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-definable functions of $ZAC^0$ are contained in $\mathsf{AC}^0_{\{|||id|||\}}$. These functions can all be computed by p-uniform log log-depth poly sized, unbounded fan-in AND, OR, NOT circuits. By Hastad [6], no log log-depth poly sized, unbounded fan-in AND, OR, NOT circuits can compute parity. The theorem then follows from Theorem 5.2 since Parity is in polynomial time and $S^1_2 \subseteq S_2$ can $\Sigma^{\mathsf{b}}_1(\mathsf{LH})$-define any polynomial-time function [3]. □

## References

[1] M. L. Bonet and T. Pitassi and R. Raz No feasible interpolation for $\mathsf{TC}^0$-Frege Proofs. In *Proceedings 38th Symposium on Foundations of Computer Science*, pages 254–263, 1997.

[2] M. L. Bonet and T. Pitassi and R. Raz Non-automatizabilty of Bounded Depth-Frege Proofs. In *Proceedings of Computational Complexity '99*, 1999.

[3] S.R. Buss. *Bounded Arithmetic.* Bibliopolis, Napoli, 1986.

[4] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetics.* Springer-Verlag, 1993.

[5] P. Clote and G. Takeuti. First order bounded arithmetic and small boolean circuit complexity classes. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 154–218. Birkhäuser, Boston, 1995.

[6] J. Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on theory of Computing*, pages 6–20, 1987.

[7] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy: a closer connection. In *Proceedings Fifth Annual Structures in Complexity Conference*, pages 169–178, 1990.

[8] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, August 1988.

[9] J. Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory.* Cambridge University Press, 1995.

[10] S.E. Mocas. Using Bounded Query Classes to Separate Classes in the Exponential Time Hierarchy from $\mathsf{PH}$. *Ph.D. Thesis*, Northeastern University. 1993.

[11] J. Johannsen. Weak Bounded Arithmetic, the Diffie-Hellman Problem and Constable's Class In *Proceedings of Logic in Computer Science 1999.*

[12] C. Pollett. Structure and definability in general bounded arithmetic theories. Annals of Pure and Applied Logic. Vol. 100. pages 189–245, October 1999.

[13] C. Pollett. Multifunction algebras and the provability of $\mathsf{PH} \downarrow$. Annals of Pure and Applied Logic. Vol. 104 July 2000. pp. 279–303.

[14] C. Pollett. Translating $I\Delta_0(exp)$ proofs into weaker systems. Mathematical Logic Quarterly. 46:246-256(No.2) May 2000.

[15] A.A. Razborov. Bounded arithmetic and lower bounds in Boolean complexity. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 344–386. Birkhauser, 1995.