# An Application of Boolean Complexity to Separation Problems in Bounded Arithmetic


Samuel R. Buss*

Department of Mathematics

University of California, San Diego

Jan Krajíček†

Mathematics Institute

Czechoslovakian Academy of Sciences, Prague


September 11, 1998


**Abstract**

We develop a method for establishing the independence of some $\Sigma_i^b(\alpha)$-formulas from $S_2^i(\alpha)$. In particular, we show that $T_2^i(\alpha)$ is not $\forall\Sigma_i^b(\alpha)$-conservative over $S_2^i(\alpha)$.

We characterize the $\Sigma_1^b$-definable functions of $T_2^1$ as being precisely the functions definable as projections of polynomial local search (PLS) problems.


Although it is still an open problem whether bounded arithmetic $S_2$ is finitely axiomatizable, considerable progress on this question has been made: $S_2^{i+1}$ is $\forall\Sigma_{i+1}^b$-conservative over $T_2^i$ [3], but it is not $\forall\Sigma_{i+2}^b$-conservative unless $\Sigma_{i+2}^p = \Pi_{i+2}^p$ [10], and in addition, $T_2^i$ is not $\forall\Sigma_{i+1}^b$-conservative over $S_2^i$


*Supported in part by NSF grants DMS-8902480 and INT-8914569. Part of this work performed while visiting the Czechoslovakian Academy of Sciences. Email address: sbuss@ucsd.edu.

†Supported in part by NSF grant INT-8914569. Part of this work performed while visiting the Univ. of Illinois, Champaign-Urbana. Email address: krajicek@csearn.bitnet.

unless $LogSpace^{\Sigma_i^p} = \Delta_{i+1}^p$ [8]. In particular, $S_2$ is not finitely axiomatizable provided that the polynomial time hierarchy does not collapse [10].

For the theory $S_2(\alpha)$ these results imply (with some additional arguments) absolute results: $S_2^{i+1}(\alpha)$ is $\forall\Sigma_{i+1}^b(\alpha)$-conservative but not $\forall\Sigma_{i+2}^b(\alpha)$-conservative over $T_2^i(\alpha)$, and $T_2^i(\alpha)$ is not $\forall\Sigma_{i+1}^b(\alpha)$-conservative over $S_2^i(\alpha)$. Here $\alpha$ represents a new uninterpreted predicate symbol adjoined to the language of arithmetic which may be used in induction formulas; from a computer science perspective, $\alpha$ represents an oracle.

In this paper we pursue this line of investigation further by showing that $T_2^i(\alpha)$ is also not $\forall\Sigma_i^b(\alpha)$-conservative over $S_2^i(\alpha)$. This was known for $i = 1, 2$ by [9, 17], see also [2], and our present proof uses a version of the pigeonhole principle similar to the arguments in [2, 9].

Perhaps more importantly, we formulate a general method (Theorem 2.6) which can be used to show the unprovability of other $\Sigma_i^b(\alpha)$-formulas from $S_2^i(\alpha)$. We demonstrate this by showing that an iteration principle, a $\Sigma_i^b(\alpha)$-formula, is also unprovable in $S_2^i(\alpha)$. This iteration principle is provable in $T_2^i(\alpha)$.

Our methods are analogous in spirit to the proof strategy of [8]: prove a witnessing theorem to show that provability of a $\Sigma_{i+1}^b(\alpha)$-formula $A$ in $S_2^i(\alpha)$ implies that it is witnessed by a function of certain complexity and then employ techniques of boolean complexity to construct an oracle $\alpha$ such that the formula $A$ cannot be witnessed by a function of the prescribed complexity. Our formula $A$ shall be $\Sigma_i^b(\alpha)$ and thus we can use the original witnessing theorem of [2]. The boolean complexity used is the same as in [8], namely Hastad's switching lemmas [6].

Johnson, Papadimitriou and Yannakakis [7] introduced a class of polynomial local search (PLS) problems. In the final section of this paper, we provide a characterization of the $\Sigma_1^b$-definable (multivalued) functions of $T_2^1$, by showing that for any PLS problem $L$, the existence of local optima for $L$ can be expressed as a $\forall\Sigma_1^b$ formula provable in $T_2^1$, and conversely, by showing that every $\forall\Sigma_1^b$-formula provable in $T_2^1$ can be witnessed by a function which is a projection of a PLS problem.

We assume the reader is familiar with bounded arithmetic and with the basics of boolean complexity. A reference on boolean complexity is [6] and on bounded arithmetic is [2] or the broader survey in the monograph [5]. The boolean circuits used in this paper are always constructed with unbounded fanin AND's and OR's in alternating levels; NOT gates are not used, instead input signals $p$ may be negated (denoted $\overline{p}$).

# 1 Some Boolean Complexity

(1.1) For $k$, $m \geq 1$, $i \geq 0$ we shall consider the set $B_{k,i}(m)$ of $m^{k+i}$ Boolean variables $p_{x_1,\dots,x_k,y_1,\dots,y_i}$, where $0 \leq x_1,\dots,x_k,y_1,\dots,y_i < m$. The set $B_{k,i}(m)$ is partitioned into $m^{k+i-1}$ blocks $(B_{k,i}(m))_j$ of the form $(B_{k,i}(m))_j = \{p_{x_1,\dots,x_k,y_1,\dots,y_{i-1},z} | z < m\}$; where $j$ is the tuple $\langle x_1,\dots,x_k,y_1,\dots,y_{i-1}\rangle$. We shall henceforth use $\vec{x}$ as an abbreviation for $x_1,\dots,x_k$. Note that $B_{k,0}(m)$ is the set of variables $p_{\vec{x}}$ with $\vec{x} < m$.

(1.2) A *restriction* $\rho$ is a partial truth evaluation of propositional variables, i.e., a partial map into $\{0,1\}$. Instead of saying that $\rho(p)$ is undefined we shall write $\rho(p) = *$.

(1.3) $\Sigma_j^{S,t}$ is the class of depth $(j+1)$ circuits with arbitrary variables, with top gate (level $j+1$) OR and at most $S$ gates in each of the levels $2, 3, \dots, j+1$, and with bottom gates (level 1) of arity at most $t$. Recall our convention that all circuits have unbounded fanin ANDs and ORs in alternating levels.

(1.4) $\mathbb{R}_{k,i,m}^{+}(q)$, $0 < q < 1$, is the probability space of restrictions $\rho$ defined on $B_{k,i}(m)$ as follows: for any $j$ and for any $p \in (B_{k,i}(m))_j$, $\rho(p) = s_j$ with probability $q$ and $\rho(p) = 1$ with probability $1-q$, where $s_j = *$ with probability $q$ and $s_j = 0$ with probability $1-q$.

The probability space $\mathbb{R}_{k,i,m}^{-}(q)$ is defined in the same way as $\mathbb{R}_{k,i,m}^{+}(q)$ except that the values 0 and 1 of $\rho$ are interchanged.

(1.5) For $i \geq 1$, $\eta_i$ is the map from $B_{k,i}(m)$ onto $B_{k,i-1}(m)$ defined by:

$$\eta_i(p_{\vec{x},y_1,\dots,y_i}) = p_{\vec{x},y_1,\dots,y_{i-1}}.$$

For $\rho$ in $\mathbb{R}^{+}_{k,i,m}(q)$, $g(\rho)$ is a restriction assigning value 1 to every variable $p_{\vec{x},y_1,\ldots,y_{i-1},s}$ which was given $*$ by $\rho$ such that for some $s < t < m$, the variable $p_{\vec{x},y_1,\ldots,y_{i-1},t}$ was also assigned $*$ by $\rho$. Thus $g(\rho)$ changes all but one $*$ in every block $(B_{k,i}(m))_j$ into 1 (if there were any $*$'s). If $\rho$ is from $\mathbb{R}^{-}_{k,i,m}(q)$, then the map $g(\rho)$ is defined identically using 0 instead of 1.

$\eta_i(\rho)$ is abbreviation for the composition of restrictions $\restriction g(\rho) \restriction \eta_i$. The effect of the restriction $\eta_i(\rho)$ is, in each block of variables, to rename one (if any) $*$'ed variable $p_{\vec{x},y_1,\ldots,y_i}$ to $p_{\vec{x},y_1,\ldots,y_{i-1}}$. If there are multiple $*$'ed variables in a block then only one is renamed and the rest are mapped to 1 (respectively, 0).

(1.6) The next lemma is Hastad's second switching lemma, see [6].

**Lemma**(*Hastad*) *Let $C$ be a $\Sigma^{S,t}_{j+1}$ circuit with variables from $B_{k,i}(m)$, $i,j \geq 1$, and $0 < q < 1$. Assume that a restriction $\rho$ is randomly chosen from $\mathbb{R}^{+}_{k,i,m}(q)$ or $\mathbb{R}^{-}_{k,i,m}(q)$. Then the probability that the function $(C \restriction \rho) \restriction \eta_i(\rho)$ is not computable by a $\Sigma^{S,t}_{j}$ circuit with variables from $B_{k,i-1}(m)$ is at most $S \cdot (6qt)^t$.*

The function $(C \restriction \rho) \restriction \eta_i(\rho)$ is defined in the obvious way: first partially evaluate and rename variables by $\rho$ and $\eta_i$ and then compute as $C$.

(1.7) Now we shall consider particular circuits $D^{\ell}_{i,m}(\vec{x})$ of depth $i$, one for every choice of $x_1, \ldots, x_k < m$. These circuits compute modified Sipser functions, see [6], and are defined by

$$D^{\ell}_{i,m}(\vec{x}) = \underset{y_1<m}{\mathrm{AND}} \quad \underset{y_2<m}{\mathrm{OR}} \quad \cdots \quad \underset{y_{i-1}<m}{Q^{i-1}} \quad \underset{y_i<\left(\frac{1}{2}\ell m \log(m)\right)^{1/2}}{Q^{i}} \quad p_{\vec{x},y_1,\ldots,y_i},$$

where $Q^{i-1}$ (resp. $Q^i$) is AND if $i$ is even (resp. odd) and is OR otherwise. Our logarithms are always base 2. Note that for distinct tuples $\vec{x}$, the circuits $D^{\ell}_{i,m}(\vec{x})$ contain distinct propositional variables. The parameter $\ell$ is introduced for technical reasons and its value will be fixed in the proof of Lemma 1.8.

(1.8) The next lemma is also due to Hastad [6]. As our parameters are slightly different from those in [6] we include a brief proof-sketch.

We say that circuit $C$ contains circuit $D$ if by renaming and/or erasing some variables we can transform $C$ into $D$.

**Lemma** *Let $\ell, m, i \geq 1$ and $x_1, \ldots, x_k < m$ and $D$ be $D^{\ell}_{i,m}(\vec{x})$. Let $q = \left(\frac{2\ell\log(m)}{m}\right)^{1/2}$ and assume $q \leq 1/5$. For $m$ sufficiently large, the following hold:*

*(a) Assume $i \geq 2$ and that a restriction $\rho$ is randomly chosen from $\mathbb{R}^{+}_{k,i,m}(q)$ if $i$ is odd or from $\mathbb{R}^{-}_{k,i,m}(q)$ if $i$ is even. Then the probability that $(D \restriction \rho) \restriction \eta_i(\rho)$ does not contain $D^{\ell-1}_{i-1,m}(\vec{x})$ is at most $\frac{1}{3}m^{-\ell+i-1}$.*

*(b) Assume $i = 1$ and that a restriction $\rho$ is randomly chosen from $\mathbb{R}^{+}_{k,1,m}(q)$. Then with probability at least $1 - \frac{1}{6}m^{-\ell+k}$ all $m^k$ circuits $D^{\ell}_{1,m}(\vec{x})$, for every choice of $x_1, \ldots, x_k < m$, are collapsed by $\restriction \rho \restriction \eta_1(\rho)$ to $*$ or $0$, and with probability at least $1 - \frac{1}{6}m^{-\ell+k}$, at least $((\ell-1)\log(m))^{1/2}m^{k-1/2}$ $*$'s are assigned.*

**Proof** (Sketch, see [6]): (a) assume that $i \geq 2$ is odd and $\rho$ is chosen randomly from $\mathbb{R}^{+}_{k,i,m}(q)$ (the case of $i$ even is analogous). Then a depth 2 subcircuit of $D$ is an OR of $m$ ANDs each of them of size $\left(\frac{1}{2}\ell m\log(m)\right)^{1/2}$:

$$\underset{y_{i-1}<m}{\mathrm{OR}} \quad \underset{y_i<\left(\frac{1}{2}\ell m\log(m)\right)^{1/2}}{\mathrm{AND}} \quad p_{x_1,\ldots,x_k,y_1,\ldots,y_i}.$$

Each AND corresponds to one class $(B_{k,i}(m))_j$ of the decomposition of $B_{k,i}(m)$. An AND gate takes value $s_j$ with probability at least

$$\begin{aligned} 1-(1-q)^{\left(\frac{1}{2}\ell m\log(m)\right)^{1/2}} &= 1-\left(1-\left(\frac{2\ell\log(m)}{m}\right)^{1/2}\right)^{\left(\frac{1}{2}\ell m\log(m)\right)^{1/2}} \\ &> 1-e^{-\ell\log(m)} > 1-\frac{1}{6}m^{-\ell}, \end{aligned}$$

for $m$ sufficiently large. Thus with probability at least $1 - \frac{1}{6}m^{-\ell+i-1}$ this is true of all $m^{i-1}$ ANDs on level 1.

For each depth two subcircuit (OR of ANDs), the expected number of ANDs for which the value of $s_j$ is equal to $*$ instead of 0 is:

$m \cdot q = (2\ell m \log(m))^{1/2}$, and, in fact, there are at least $(\frac{(\ell-1)m\log(m)}{2})^{1/2}$ $*$'s among $s_j$'s with probability at least $1 - \frac{1}{6}m^{-\ell}$. This is seen by the following argument:

Let $r_u$ be the probability that exactly $u$ of the $s_j$'s corresponding to ANDs of the OR gate, are equal to $*$. Then

$$r_u = \binom{m}{u} \left(\frac{2\ell \log(m)}{m}\right)^{u/2} \left(1 - \left(\frac{2\ell \log(m)}{m}\right)^{1/2}\right)^{m-u}.$$

For $u \leq (\ell m \log(m))^{1/2}$ it holds that $r_u / r_{u-1} \geq \sqrt{2}$ and, as $r_{(\ell m \log(m))^{1/2}} < 1$, we get the estimate:

$$\begin{aligned}
\sum_{u=0}^{\left(\frac{1}{2}\ell m \log(m)\right)^{1/2}} r_u &\leq r_{\left(\frac{1}{2}\ell m \log(m)\right)^{1/2}} \sum_{u=0}^{\infty} 2^{-u/2} \\
&< 4 \cdot r_{\left(\frac{1}{2}\ell m \log(m)\right)^{1/2}} \\
&\leq 4\left(\sqrt{2}\right)^{-(1-2^{-1/2})(\ell m \log(m))^{1/2}} \cdot r_{(\ell m \log(m))^{1/2}} \\
&\leq 4\left(\sqrt{2}\right)^{-(1-2^{-1/2})7\ell \log(m)} \\
&\leq \frac{1}{6} m^{-\ell},
\end{aligned}$$

for $m$ sufficiently large. (The next-to-last inequality used $m \geq 49\ell \log m$ which follows from $q \leq 1/5$.)

As there are $m^{i-2}$ ORs on level 2 at $D$, the probability that every such OR gets assigned at least $\left(\frac{1}{2}(\ell-1)m\log(m)\right)^{1/2}$ $*$'s is at least $1 - \frac{1}{6}m^{-\ell+i-2}$. This proves part (a). Part (b) is proved completely analogously.
Q.E.D. Lemma 1.8

## 2 Oracle computations of witnessing functions

(2.1) A polynomial time oracle machine $M$ is a Turing machine running in polynomial time and querying an oracle; for different oracles the machine may compute different functions. Thus we think of the machine as described independently of a specific oracle.

(2.2) A $\Sigma_i^p(\alpha)$-*oracle* machine is a pair $(M, B(x))$, where $B(x)$ is a $\Sigma_i^b(\alpha)$-formula and $M$ is a polynomial-time oracle machine. For the rest of this section, $\alpha$ is a $(k+i)$-ary predicate symbol.

For a particular predicate $\alpha \subseteq \mathbb{N}^{k+i}$, $B(x)$ defines a subset of $\mathbb{N}$, i.e.,an oracle, and $(M, B(x))$ computes a particular function. We shall denote by $M^\alpha$ machine $M$ with the oracle $B(x)$.

(2.3) A *circuit oracle* is a function $C$ assigning to each $u \in \mathbb{N}$ a boolean circuit $C_u$ with variables from some $B_{k,i}(m)$, $m = m(u)$ being a function of $u$ and $k, i$ fixed. For a particular $\alpha \subseteq \mathbb{N}^{k+i}$, the circuit oracle $C$ defines a subset $C^\alpha$ of $\mathbb{N}$ of those $u$ for which $C_u$ computes 1 when propositional variables are assigned truth values according to:

$$p_{x_1,\ldots,x_k,y_1,\ldots,y_i} = 1 \quad \text{iff} \quad (x_1, \ldots, x_k, y_1, \ldots, y_i) \in \alpha.$$

For $M$ an oracle Turing machine and $\alpha \subseteq \mathbb{N}^{k+i}$, we let $M^\alpha$ denote the machine $M$ using the oracle $C^\alpha$. The context will always distinguish between the two definitions (2.2) and (2.3) of $M^\alpha$.

For $S, t$ and $m$ functions of $u$, a circuit oracle is called $\Sigma_j^{S,t}$-circuit oracle with variables from $B_{k,i}(m)$ if $C_u$ is a $\Sigma_j^{S(u),t(u)}$-circuit with variables from $B_{k,i}(m(u))$ for all $u$.

There is a close correspondence between the $\Sigma_i^b(\alpha)$-oracles and $\Sigma_i^{S,t}$-circuit oracles with variables from $B_{k,i}(m)$, with $S = 2^{(\log m)^c}$, $t = \log S$ and $m = 2^{(\log u)^c}$ (see [4]). Namely, if $(M, B(x))$ is as in (2.2), then the oracle $B(x)$ is equivalent to a family of $\Sigma_i^{S,t}$ circuits $C_u$ with variables from $B_{k,i}(m)$, with $S, t, m$ bounded as above for some constant $c$ depending on the runtimes of $M$ and $B$. As $B(x) \in \Sigma_i^b$, $B(x)$ may be computed by making $i$ blocks of existential/universal guesses and then running for polynomial time. Hence, for each $u$, a $\Sigma_i^{S,t}$ circuit $C_u$ with variables from $B_{k,i}(m)$ ($m = 2^{(\log u)^{O(1)}}$) may be constructed that computes $B(u)$ by letting $i$ levels of OR's and AND's in $C_u$ correspond to blocks of existential and universal guesses, respectively, and at the bottom of the circuit, expressing a polynomial time execution of $B$ (performed after all nondeterministic choices are finished), as either an OR of AND's of fanin $\leq t$ or an AND of OR's of fanin $\leq t$ (if $i$ is odd or even, respectively). Merging adjacent OR's (respectively, AND's) in the second and third levels from the bottom of the circuit, makes $C_u$ have depth $i+1$ as desired.

Thus any $\Sigma_i^b(\alpha)$-oracle may be viewed as a $\Sigma_i^{S,t}$-circuit oracle with variables from $B_{k,i}(m)$ and $S, t, m$ bounded in terms of $u$ as above. The converse is not true; however, any such $\Sigma_i^{S,t}$-circuit oracle may nonetheless be viewed as analogous to a *non-uniform* $\Sigma_i^b(\alpha)$-oracle.

(2.4) Fix $m$; let $[m]$ denote the set $\{0, 1, \ldots, m-1\}$. A $(k-u)$-dimensional cylinder in $[m]^k$ is any set of the form:

$$\{(x_1, \ldots, x_k) \in [m]^k : x_{i_1} = a_1, \ldots, x_{i_u} = a_u\}$$

for any fixed values $i_1 < \ldots < i_u$ and $a_1, \ldots, a_u < m$. There are $\binom{k}{r} m^{k-r}$ many $r$-dimensional cylinders in $[m]^k$.

(2.5) For $\alpha$ a $k+i$-ary predicate, denote by $A^{i,\alpha}(a, x_1, \ldots, x_k)$ the $\Pi_i^b(\alpha)$-formula:

$$\forall y_1 < a \, \exists y_2 < a \cdots Q y_{i-1} < a \; Q' y_i < \left(\tfrac{1}{2} \ell a \log a\right)^{1/2} \alpha(x_1, \ldots, x_k, y_1, \ldots, y_i).$$

Thus $A^{i,\alpha}$ has $(k+1)$ free variables. The parameter $\ell$ relates to $\ell$ in (1.7) and its value will be fixed later.

Let $\beta(x_1, \ldots, x_k)$ be a $k$-ary predicate symbol and let $B(a, \beta)$ be a bounded formula containing $\beta$ in which $a$ is the only free variable, in which every quantifier is bounded by $a$, which contains no function symbols, and in which every occurence of $\beta$ has $k$ bound variables as arguments. Obviously, $B(a, \beta)$ depends only on the values of $\beta(a_1, \ldots, a_k)$ where $a_1, \ldots, a_k < a$. Define $B(a, A^{i,\alpha})$ to be the $\Sigma_\infty^b(\alpha)$ formula obtained from $B(a, \beta)$ by replacing all occurences of $\beta(x_1, \ldots, x_k)$ by $A^{i,\alpha}(a, x_1, \ldots, x_k)$.

We shall assume $B$ begins with an existential quantifier, so $B$ is $\exists\, x < a\, D$. A witness for $B(a, \ldots)$ is a value for $z$ such that $D(a, z, \ldots)$ holds. We shall see examples of such formulas in the next section.

(2.6) The next theorem is the main technical result of this paper.

**Theorem** *Assume $i, k \geq 1$ and that $\alpha$, $A^{i,\alpha}$ and $B(a, \beta)$ are as in (2.5). Assume also that $M$ is a polynomial time oracle machine with a $\Sigma^b_{i+1}(\alpha)$-oracle, such that for every $\alpha \subseteq \mathbb{N}^{k+i}$ the machine $M^\alpha$ computes from input $a$ some witness to formula $B(a, A^{i,\alpha})$.*

*Then there is a constant $c \geq 1$ such that for $m$ sufficiently large there is a $Q \subseteq \mathbb{N}^k$ and a $\Sigma^{S,t}_1$-circuit oracle $C^1$ with variables from $B_{k,0}(m)$ so that the following conditions hold:*

(i) *for all $u$, $m(u) = m, S(u) = 2^{(\log m)^c}$ and $t(u) = \log S = (\log m)^c$,*

(ii) *for every $r$-dimensional cylinder $U$ in $[m]^k$, $r = 1, \ldots, k$,*

$$|U \backslash Q| \geq m^{r-1/2}$$

(iii) *for every $\alpha^0 \subseteq \mathbb{N}^k$ s.t. $\alpha^0 \cap Q = \emptyset$, machine $M^{\alpha^0}$ computes on input $m$ a witness to formula $B(m, \alpha^0)$.*

Note that for any given $m$, $S(u), t(u)$ and $m(u)$ are constants independent of $u$ and that the variables of the $\Sigma^{S,t}_1$-circuit oracle are of the form $p_{x_1,\ldots,x_k}$, with $x_1, \ldots, x_k < m$, and thus $M^{\alpha^0}$ is correctly defined for any $\alpha^0 \subseteq \mathbb{N}^k$.

To better understand Theorem 2.6, first consider a converse of it: if $N$ is a Turing machine which, given an input $m$ and given a $\Sigma^{S,t}_1$-circuit oracle $C$ involving variables $p_{\vec{x}}$, always outputs a witness for $B(m, \alpha^0)$, then the same Turing machine $N$ can find a witness for $B(m, A^{i,\alpha})$ when given $m$ as input and given a $\Sigma^{S,t}_{i+1}$-circuit oracle $C'$ with variables from $B_{k,i}(m)$. This converse is easily proved if $C'$ is defined from $C$ by replacing variables $p_{\vec{x}}$ by $\Pi^{S,t}_i$ subcircuits for $A^{i,\alpha}$ (with variables from $B_{k,i}(m)$)— note that in $C$, variables $p_{\vec{x}}$ give the truth values of $\alpha^0(\vec{x})$, while in $C'$, variables $p_{\vec{x},y_1,\ldots,y_i}$ give truth values of $\alpha(\vec{x}, y_1, \ldots, y_i)$.

Since a $\Sigma^b_{i+1}(\alpha)$-oracle can be translated into a $\Sigma^{S,t}_{i+1}$-circuit oracle with variables from $B_{k,i}(m)$, Theorem 2.6 essentially states that the converse can be partially reversed, at least for $\alpha^0$'s that avoid the set $Q$. The set $Q$ is small in the sense that, in any cylinder, at least a fraction $1/\sqrt{m}$ of the $k$-tuples from the cylinder are not in $Q$ (and hence may be $\alpha^0$).

Another way to think about Theorem 2.6 is as follows: Suppose there is a machine $M$ that finds witnesses for $B(a, A^{i,\alpha})$ with a $\Sigma^b_{i+1}(\alpha)$-oracle. Since $A^{i,\alpha}$ is a $\Pi^b_i$-formula, $M$ has the power to ask existential questions involving $A^{i,\alpha}$. The point of Theorem 2.6 is that $M$ does not have very much more power; namely, if $M$ asks instead $\Sigma_1^{S,t}$-circuit oracle queries about $\beta$, then $M$ can find a witness for $B(a,\beta)$ for many $\beta$'s (the ones that avoid $Q$).

**Proof** of the theorem: The proof consists of several steps, employing heavily the lemmas from section 1.

1. Choose $m$ sufficiently large so that Lemma 1.8 holds and fix $a = m$. Let $\ell \geq i + 2k$.

2. For technical reasons (Lemma 1.8), we forbid into $\alpha$ any members $(x_1,\ldots,x_k,y_1,\ldots,y_i)$ with $x_1,\ldots,x_k,y_1,\ldots,y_{i-1} \geq m$ or with $y_i \geq (\frac{1}{2}\ell m \log(m))^{1/2}$; this can be done without loss of generality because of the form of the bounded quantifiers in $B$ and in $A^{i,\alpha}$. Clearly the truth value of $A^{i,\alpha}(a,x_1,\ldots,x_k)$ is computed by the circuit $D^\ell_{i,m}(x_1,\ldots,x_k)$ under the evaluation of variables:

$$p_{x_1,\ldots,x_k,y_1,\ldots,y_i} = 1 \quad \text{iff} \quad (x_1,\ldots,x_k,y_1,\ldots,y_i) \in \alpha.$$

3. Let $E(x)$ be the $\Sigma^b_{i+1}(\alpha)$-oracle of the oracle machine. Since the machine $M$ is polynomial time, any number $u$ occuring in the computation is bounded by $2^{(\log m)^{O(1)}}$. For any $u < 2^{(\log m)^{O(1)}}$, the truth value of $E(u)$ is computed by a $\Sigma^{S,t}_{i+1}$-circuit $C_u$ with variables from $B_{k,i}(m)$, where $S \leq 2^{(\log m)^c}$ and $t = \log(S)$, for $c$ large enough.

   Thus we henceforth think of $M$ as being a $\Sigma^{S,t}_{i+1}$-circuit–oracle (with variables from $B_{k,i}(m)$) machine with $S,t,m$ constants.

4. Let $\rho_j$ be randomly chosen restrictions from $\mathbb{R}^{\epsilon_j}_{k,j,m}(q_j)$, for $j = i, i-1, \ldots, 1$, where $\epsilon_j$ is $+$ if $j$ is odd and $-$ if $j$ is even, and $q_j = \left(\frac{2(\ell-i+j)\log(m)}{m}\right)^{1/2}$. We are interested in what the effect of the composed restriction

$$\kappa = \restriction \rho_i \restriction \eta_i(\rho_i) \restriction \rho_{i-1} \restriction \eta_{i-1}(\rho_{i-1}) \restriction \ldots \restriction \rho_1 \restriction \eta_1(\rho_1)$$

   is on the circuits $C_u$ and $D^\ell_{i,m}(x_1,\ldots,x_k)$

5. By (1.8), any circuit $D^{\ell+j-i}_{j,m}(x_1,\ldots,x_k)$ contains, after being restricted by $\rho_j \restriction \eta_j(\rho_j)$, the circuit $D^{\ell+j-i-1}_{j-1,m}(x_1,\ldots,x_k)$ with probability at least $1 - \frac{1}{3}m^{-\ell+i-1}$, and thus this is true for all $m^k$ circuits $D^{\ell+j-1}_{j,m}(x_1,\ldots,x_k)$

obtained by considering all values of $x_1, \ldots, x_k < m$, with probability at least

$$1 - \frac{1}{3} m^{-\ell+k+i-1}.$$

6. Applying successively the restrictions $\rho_j \restriction \eta_j(\rho_j)$, with $j = i, \ldots, 2$, to $D^{\ell}_{i,m}(x_1, \ldots, x_k)$, transforms the circuit into

$$(D^{\ell}_{i,m}(x_1, \ldots, x_k)) \restriction \rho_i \restriction \eta_i(\rho_i) \restriction \ldots \restriction \rho_2 \restriction \eta_2(\rho_2),$$

   and therefore, by the preceeding paragraph, with probability at least

$$1 - \frac{1}{3}(i-1) m^{-\ell+k+i-1}$$

   each of these $m^k$ circuits contains the circuit $D^{\ell-i+1}_{1,m}(x_1, \ldots, x_k)$.

7. To establish condition (ii) of the theorem, we have to be more careful in assessing what happens to $D^{\ell-i+1}_{1,m}(x_1, \ldots, x_k)$ after being restricted by the randomly chosen $\restriction \rho_1 \restriction \eta_1(\rho_1)$.

   Let $U$ be any $r$-dimensional cylinder; $r \geq 1$. Then analogously to part (b) of Lemma 1.8 and by reasoning similar to the proof of Lemma 1.8(a), with probability at least

$$1 - \frac{1}{6} m^{-\ell+i-1+r},$$

   there are at least

$$m^r \left( \frac{(\ell - i)\log(m)}{m} \right)^{1/2} \geq m^{r-1/2}$$

   many $*$'s assigned to the $m^r$ many circuits corresponding to $(x_1, \ldots, x_k) \in U$. At the same time, with probability at least

$$1 - \frac{1}{6} m^{-\ell+i-1+r}$$

   none of these circuits collapses to 1. Summing up, with probability at least

$$1 - \frac{1}{3} m^{-\ell+i-1+r},$$

all $m^r$ circuits corresponding to $(x_1, \ldots, x_k) \in U$ collapse to either $*$ (i.e. to $p_{x_1,\ldots,x_k}$) or to $0$, with at most $m - m^{r-1/2}$ collapsing to $0$.

Counting over all cylinders of dimension $\geq 1$, the above holds for all such cylinders $U$ with probability at least

$$\begin{aligned} 1 - \sum_{r=1}^{k} (\frac{1}{3} m^{-\ell+i-1+r} m^{k-r} \binom{k}{r}) &= 1 - \frac{1}{3} m^{-\ell+i-1+k} \sum_{r=1}^{k} \binom{k}{r} \\ &\geq 1 - \frac{1}{3} m^{-\ell+i-1+k} 2^k \\ &\geq 1 - \frac{1}{3} m^{-\ell+i-1+2k}. \end{aligned}$$

8. Now we turn our attention to what effect $\kappa$ has on the oracle circuits $C_u$.

   By Lemma 1.6, any $\Sigma_{j+1}^{S,t}$ circuit with variables from $B_{k,j}(m)$ is transformed by the restriction $\restriction \rho_j \restriction \eta_j(\rho_j)$ into a $\Sigma_j^{S,t}$-circuit with variables from $B_{k,j-1}(m)$ with probability at least

$$1 - S(6q_j t)^t.$$

   Therefore with probability at least

$$1 - S(6t)^t \left( \sum_{j=i}^{1} q_j^t \right) \geq 1 - S(6t)^t \cdot i \cdot (q_i)^t$$

   (since $q_i \geq q_{i-1} \geq \ldots \geq q_1$), a $\Sigma_{i+1}^{S,t}$ circuit $C_u$ with variables from $B_{k,i}(m)$ is transformed by $\kappa$ into a $\Sigma_1^{S,t}$-circuit $C_u^1$ with variables from $B_{k,0}(m)$. It follows that with probability at least

$$1 - S^2 \cdot i \cdot (6q_i t)^t$$

   $C_u^1 = C_u \restriction \kappa$ is a $\Sigma_1^{S,t}$-circuit, for all $u < S$. In other words, every circuit oracle that $M$ may query collapses to a $\Sigma_1^{S,t}$ with this probability. It is easy to compute that for $m$ large enough (w.r.t. $\ell$ and $c$):

$$1 - S^2 \cdot i \cdot (6q_i t)^t \geq 1 - 2^{-\frac{1}{3}\log(m)^{c+1}}.$$

9. By 6. and 7., $\kappa$ collapses every $D^{\ell}_{i,m}(x_1, \dots, x_k)$ into $p_{x_1,\dots,x_k}$ or $0$, with "cylinder property" of 7. satisfied, with probability at least

$$1 - \frac{1}{3}(i-1)m^{-\ell+k+i-1} - \frac{1}{3}m^{-\ell+i-1+2k} \;\geq\; 1 - \frac{i}{3}m^{-\ell+i-1+2k}.$$

By 8., with probability at least

$$1 - 2^{-\frac{1}{3}\log(m)^{c+1}},$$

every $C^1_u = C_u \restriction \kappa$ is a $\Sigma^{S,t}_1$-circuit with variables from $B_{k,0}(m)$.

Thus both these events happen, for random $\kappa = \rho_i \restriction \eta_i(\rho_i) \restriction \dots \restriction \rho_1 \restriction \eta_1(\rho_1)$, with probability at least:

$$1 - \frac{i}{3}m^{-\ell+i-1+2k} - 2^{-\frac{1}{3}\log(m)^{c+1}} \;\geq\; 1 - \frac{i}{3m} - \frac{1}{8} \;\geq\; \frac{1}{2},$$

since $\ell \geq i + 2k$, for $m$ large enough.

10. By 9., there is at least one $\kappa$ satisfying conditions at 8. Define

$$Q = \{(x_1, \dots, x_k) \mid D^{\ell}_{i,m}(x_1, \dots, x_k) \restriction \kappa = 0\}.$$

$Q$ satisfies property (ii) of Theorem 2.6 by 7.

Define the $\Sigma^{S,t}_1$-circuit oracle by

$$C^1_u := C_u \restriction \kappa.$$

Now, condition (iii) of Theorem 2.6 is satisfied by construction.

Q.E.D. Theorem 2.6

(2.7) Observe that the above proof works even if $S$ is considerably larger: up to $S = 2^{m^{(\frac{1}{2}-\epsilon)}}$, $\epsilon > 0$ fixed. In other words, we can allow the machine $M$ to run in time $2^{m^{(\frac{1}{2}-\epsilon)}}$. The only modification to the proof is to the calculations in 8., recall that $t = \log S$.

## 3 The Pigeonhole Principle

In this section we apply Theorem 2.6 to show the unprovability of a weak form of the pigeonhole principle in $S_2^i(\alpha)$.

(3.1) Let $\beta(x_1, x_2, x_3)$ be a predicate symbol. Let $WPHP(a, \beta)$ be the formula:

$$
\begin{aligned}
&(\forall u_1, u_2, v_1, v_2, w < a)[(\beta(u_1, u_2, w) \wedge \beta(v_1, v_2, w)) \rightarrow (u_1 = v_1 \wedge u_2 = v_2)] \\
&\quad \wedge (\forall u_1, u_2, v, w < a)[(\beta(u_1, u_2, v) \wedge \beta(u_1, u_2, w)) \rightarrow v = w] \\
&\qquad\qquad \rightarrow (\exists u_1, u_2 < a)(\forall v < a)(\neg\beta(u_1, u_2, v)).
\end{aligned}
$$

If we think of a pair of numbers $x_1$, $x_2 < a$ as coding a single number $< a^2$, then the formula $WPHP$ says that $\beta(x_1, x_2, x_3)$ does not define the graph of a one-to-one function from $a^2$ to $a$. Clearly $WPHP$ is $\Sigma_2^b(\beta)$-formula.

(3.2) Let $\alpha(x_1, x_2, x_3, y_1, \ldots, y_i)$ be a $(i+3)$-ary predicate symbol and $A^{i,\alpha}(a, x_1, x_2, x_3)$ be the $\Pi_i^b(\alpha)$-formula defined in (2.5). Then we have:

**Theorem***(Paris-Wilkie-Woods) For all $i \geq 0$, $WPHP(a, A^{i,\alpha})$ is provable by $T_2^{i+2}(\alpha)$.*

**Proof** In [15] it was shown that $WPHP(a, \beta)$ is provable in $I\Delta_0(\beta) + \Omega_1$, and thus also in $T_2(\beta)$. Already [2] has observed that this proof can be carried out in $T_2^2(\beta)$. This implies the theorem.
Q.E.D. Theorem 3.2

(3.3) **Theorem** *Let $i \geq 0$. The $\Sigma_{i+2}^b(\alpha)$-formula $WPHP(a, A^{i,\alpha})$ is not provable in $S_2^{i+2}(\alpha)$.*

**Proof** Case $i = 0$ was proved in [9]. We use Theorem 2.6 to essentially reduce the case $i > 0$ to the case $i = 0$ (we include the $i = 0$ argument here too).

Let $i \geq 1$ and assume that $S_2^{i+2}(\alpha)$ proves $WPHP(a, A^{i,\alpha})$. Then by the "main theorem" for bounded arithmetic [2], the formula $WPHP(a, A^{i,\alpha})$ is witnessed by a $\square_{i+2}^P(\alpha)$-function, i.e., by a function which is computed by a polynomial time oracle machine $M$ with a $\Sigma^b_{i+1}(\alpha)$-oracle $E(x)$. We shall consider only $\alpha$'s such that $A^{i,\alpha}$ defines a partial 1-1 function from $a^2$ to $a$; in other words, such that

$$(\forall u_1, u_2, v_1, v_2, w < a)[(A^{i,\alpha}(a, u_1, u_2, w) \wedge A^{i,\alpha}(a, v_1, v_2, w)) \rightarrow (u_1 = v_1 \wedge u_2 = v_2)]$$
$$\wedge(\forall u_1, u_2, v, w < a)[(A^{i,\alpha}(a, u_1, u_2, v) \wedge A^{i,\alpha}(a, u_1, u_2, w)) \rightarrow v = w]$$

For such $\alpha$'s, $M^\alpha$ on input $a$, will witness the truth of $WPHP(\alpha, \beta)$ by producing as output values $u_1, u_2 < a$ such that

$$(\forall v < a)(\neg A^{i,\alpha}(a, u_1, u_2, v);$$

in other words, $M^\alpha$ outputs values for $u_1, u_2$ such that the partial function defined by $A^{i,\alpha}$ is undefined at the pair $u_1, u_2$.

We now apply Theorem 2.6 with $B(a, \beta)$ the $\Sigma^b_2$-formula which is the prenex form of $WPHP(a, \beta)$. Theorem 2.6 implies, for all $m$ sufficiently large, there is a $\Sigma^{S,t}_1$-circuit oracle, $C^1_u$, with variables from $B_{3,0}(m)$, where $S = 2^{\log(m)^c}$ and $t = \log(m)^c$, and there is a $Q \subseteq [m]^3$ such that whenever $\alpha^0 \subseteq [m]^3$ and $\alpha^0 \cap Q = \emptyset$, the machine $M^{\alpha^0}$ outputs a witness to $B(m, \alpha^0)$.

We show that this is impossible. To prove this, we shall build an oracle $\alpha^0$ for which $M^{\alpha^0}(a)$ fails to witness $B(m, \alpha^0)$ — the oracle is constructed by executing $M^\alpha(m)$ and creating sets $X^+_i$ and $X^-_i$ at the $i$-th oracle query. The set $X^+_i$ (respectively, $X^-_i$) is a set of triples that is forced to be in $\alpha^0$ (respectively, out of $\alpha^0$). Initially, $X^+_0 = \emptyset$ and $X^-_0 := Q$. Let "$C^1_{u_1}$?" be the first circuit-oracle query. There are two possibilities:

(a) There is $\alpha \subseteq [m]^3$, $X^+_0 \subseteq \alpha$, $\alpha \cap X^-_0 = \emptyset$, such that $\alpha$ is a graph of partial $1-1$ map from $m^2 (= m \times m)$ to $m$, and $C^1_{u_1}$ evaluates to $1$,

(b) There is no $\alpha$ satisfying (a).

In Case (a), since $C^1_{u_1}$ is a $\Sigma^{S,t}_1$-circuit, it is an OR of AND's of size $\leq t$; thus, $C^1_{u_1}$ can be forced true by specifying the the truth values $\leq t = (\log m)^c$ atoms. Choose any partial evaluation $\xi$ that forces $C^1_{u_1}$ true such that $\xi$ sets $\leq t$ values and is consistent with conditions in (a). Form $X^+_1$ (respectively, $X^-_1$) by adding to $X^+_0$ (respectively, to $X^-_0$) all $(x_1, x_2, x_3)$ such that $p_{x_1,x_2,x_3}$ given value $1$ (respectively, value $0$) by $\xi$. Now answer YES to the machine and resume its computation.

In Case (b) put $X_1^+ := X_0^+$, $X_1^- := X_0^-$, answer NO, and resume the computation.

Arriving at $(i+1)$-st query, we have already defined $X_i^+$, $X_i^-$ so that

$$|X_i^+| \le i(\log m)^c, \qquad |X_i^- \backslash Q| \le i(\log m)^c,$$

and $X_i^+ \cap X_i^- = \emptyset$, and the answers to the first $i$ oracle queries have been fixed, for any graph $\alpha$ of a partial 1-1 function with $X_i^+ \subseteq \alpha$ and $\alpha \cap X_i^- = \emptyset$. Form $X_{i+1}^{\pm}$ analogously as above.

At the end of computation (which has $\le (\log m)^c$ steps), we define

$$X^+ = \bigcup_i X_i^+, \qquad X^- = \bigcup_i X_i^-$$

and then we have that

$$|X^+| \le (\log m)^{2c}, \qquad |X^- \backslash Q| \le (\log m)^{2c}$$

with $Q \subseteq X^-$. Furthermore, for all partial 1-1 maps $\alpha$ such that $\alpha \supseteq X^+$ and $\alpha \cap X^- = \emptyset$, the oracle queries of $M^\alpha(a)$ are fixed and thus the output $(u_1, u_2)$ of $M^\alpha$ is the same; in other words, there is a fixed output $(u_1, u_2)$ which witnesses $WPHP(m, \alpha(x_1, x_2, x_3))$ for all such $\alpha$. But this is impossible: since $Q$ was chosen to satisfy Theorem 2.6(ii), there are at least $m^{1/2}$ $v$'s such that $(u_1, u_2, v) \notin Q$, and thus at least $(m^{1/2} - (\log m)^{2c}) \ge 1$ such $v$'s not in $X^-$. Hence we can set $\alpha = X^+ \cup \{(u_1, u_2, v)\}$ for some $v$ such that $\alpha \cap X^- = \emptyset$, but obviously $(u_1, u_2)$ then does not witness $WPHP(m, \alpha)$.
Q.E.D. Theorem 3.3

(3.4) **Corollary** *$T_2^i(\alpha)$ is not $\forall\Sigma_i^b(\alpha)$-conservative over $S_2^i(\alpha)$, $i \ge 1$. Actually, $T_2^i(\alpha)$ is not $\forall\Sigma_i^b(\alpha)$-conservative over any $S_j^i(\alpha)$, $i \ge 1$, $j \ge 2$.*

**Proof** The corollary follows from Theorems 3.2 and 3.3.

Use Remark (2.7) for the second part.
Q.E.D. Corollary 3.4

The second part of Corollary 3.4 complements [12] where it was shown that $T_{j+1}^i$ is not $\Pi_1^b$-conservative over $T_j^i$, $i$, $j \ge 1$.

# 4 The Iteration Principle

(4.1) The previous section showed that $T^i_2$ is not $\forall\Sigma^b_i(\alpha)$-conservative over $S^i_2(\alpha)$ by reducing — via Theorem 2.6 — the general case $i > 2$ to the base case which is essentially equivalent to the case where $i = 2$. In this section, we give a example of another proof of the same result; the most important novel feature, is that now the base case corresponds to $i = 1$. For this, we need to prove a useful analogue of Theorem 2.6.

(4.2) A $\Delta^{S,t}_1$-circuit $C$ with variables from $B_{k,0}(m)$ is a pair of $\Sigma^{S,t}_1$-circuits $C^+$ and $C^-$ with variables from $B_{k,0}(m)$ such that $C^+$ by definition computes the value of the $\Delta^{S,t}_1$-circuit and $C^-$ must compute its negation.

A $\Delta^{S,t}_1$-circuit oracle with variables from $B_{k,0}(m)$ is a family of $\Delta^{S,t}_1$-circuits with variables from $B_{k,0}(m)$, one for each oracle query, analagously to the definitions of (2.3). $S, t$ and $m$ may depend on $u$ as before.

(4.3) **Theorem** *Assume $i, k \geq 1$ and that $\alpha(\vec{x}, \vec{y})$, $A^{i,\alpha}$ and $B(a, \beta)$ are as in (2.5). Assume also that $M$ is a polynomial time oracle machine with a $\Sigma^b_i(\alpha)$-oracle, such that for every $\alpha \subseteq \mathbb{N}^{k+i}$ the machine $M^\alpha$ computes from input $a$ some witness to the formula $B(a, A^{i,\alpha})$.*

*Then there is a constant $c \geq 1$ such that for $m$ sufficiently large there is a $Q \subseteq \mathbb{N}^k$ and a $\Delta^{S,t}_1$-circuit oracle $C$ with variables from $B_{k,0}(m)$ so that the following conditions hold:*

(i) *for all $u, m(u) = m, S(u) = 2^{(\log m)^c}$ and $t(u) = \log S = (\log m)^c$,*

(ii) *for every $r$-dimensional cylinder $U$ in $[m]^k$, $r = 1, \ldots, k$,*

$$|U \backslash Q| \geq m^{r-1/2}$$

(iii) *for every $\alpha^0 \subseteq \mathbb{N}^k$ s.t. $\alpha^0 \cap Q = \emptyset$, machine $M^{\alpha^0}$ computes on input $m$ a witness to formula $B(m, \alpha^0)$. (Recall that $M^{\alpha^0}$ operates with the circuit oracle $C^{\alpha^0}$ instead of the original $\Sigma^b_i$-oracle.)*

The difference between Theorems 2.6 and 4.3 that the former assumes $M$ has a $\Sigma^b_{i+1}$ oracle and states the existence of a $\Sigma^{S,t}_1$-circuit oracle, whereas the latter assumes $M$ has a $\Sigma^b_i$ oracle and states the existence of a $\Delta^{S,t}_1$-circuit oracle. Having a $\Delta^{S,t}_1$-circuit oracle is analogous to having only an oracle for (a polynomial time function of) $\alpha$, in the same way that having a $\Sigma_1^{S,t}$-circuit oracle was analogous to having a $\Sigma_1^b$-oracle. More precisely, when we construct an $\alpha$ by simulating $M$ with a $\Delta_1^{S,t}$-circuit, if an oracle query answer has not yet been forced, then it will always be possible to force the oracle query to a desired Yes/No answer by setting only a relatively small number (namely, $\leq t$) many values of $\alpha$. This is because both $C^{\alpha,+}$ and its complement $C^{\alpha,-}$ are OR's of small AND's; and thus either a Yes or No answer may be forced by setting values of $\alpha$ to make one AND true in $C^{\alpha,+}$ or in $C^{\alpha,-}$ (respectively).

**Proof** The proof of Theorem 4.3 is nearly identical to the proof of Theorem 2.6 except for the last step. Before the last step of the proof, $A^{i,\alpha}$'s have been collapsed to circuits consisting a single AND gate, and the $\Sigma_i^b$-oracle has been collapsed to a $\Sigma_1^{S,t}$-oracle $C^1$ (with variables from $B_{k,1}(m)$ in this case).

After one more random restriction (from $\mathbb{R}^+_{k,1,m}$) the AND gates of the the $A^{i,\alpha}$'s collapse, with high probability, to 0 or to $p_{\vec{x}}$ with the cylinder property (ii) valid, exactly as in the proof of Theorem 2.6; It remains to consider what this final restriction does to the circuit $C^1$: since $C^1$ is a family of $\Sigma_1^{S,t}$-circuits, it certainly remains one after being restricted; in addition, by the switching lemma (Lemma 1.6), its complement $\neg C^1$ becomes, with high probability, a family of $\Sigma_1^{S,t}$-circuits too. In other words, after the final restriction, the circuit oracle becomes a $\Delta_1^{S,t}$-circuit oracle with variables from $B_{k,0}(m)$.

The computations of the probabilities are identical to the proof of Theorem 2.6.
Q.E.D. Theorem 4.3

(4.4) We shall consider an *iteration principle* $Iter_0(f,a)$ which states
"If $f$ satisfies the three conditions

(1) $0 < f(0)$,

(2) $\forall x < a, f(x) = a \vee f(x) < f(f(x))$, and

(3) $\forall x < a, f(x) \leq a$,

then there exists a $b < a$ such that $f(b) = a$".

Note that $Iter_0(f,a)$ is expressible by a $\Sigma_1^b$-formula.

**Theorem** *The formula* $(\forall x) Iter_0(f,x)$ *is provable in* $T_2^1(f)$ *but not in* $S_2^1(f)$.

**Proof** To see that $T_2^1 \vdash Iter_0(f,a)$, let $\varphi(u)$ be the $\Sigma_1^b$-formula

$$(\exists x \leq u)(u < f(x) \wedge f(x) \leq a).$$

Then clearly, $T_2^1(f)$ proves $\varphi(0)$ by (1) of the definition of $Iter_0$. Also, $T_2^1(f)$ proves

$$u \leq a - 2 \wedge \varphi(u) \rightarrow \varphi(u+1);$$

to prove this, note that if $x_u$ witnesses $\varphi(u)$ then either $f(x_u) = u+1$ or $f(x_u) > u+1$, and in the former case, $f(f(x_u))$ is witness for $\varphi(u+1)$, and in the latter case, $x_u$ is already a witness for $\varphi(u+1)$. Now, by $\Sigma_1^b$-IND, $T_2^1(f)$ can prove that $\varphi(a-1)$ holds and a witness $b$ for $\varphi(a-1)$ must satisfy $f(b) = a$.

Now, for the sake of a contradiction, assume $S_2^1(f) \vdash Iter_0(f,a)$. Then there is a polynomial time Turing machine with an oracle for the function $f$ such that, on input $a$, if $f$ satisfies conditions (1)-(3) of the definition of $Iter_0$, then $M$ outputs a value $b < a$ so that $f(b) = a$. We prove this is impossible by constructing an $f$ for which $M$ fails.

For fixed $M$, take $a$ sufficiently large and start the computation of $M$ on $a$. After the $i$-th oracle query of $M$, we will have values $0 = r_0 < r_1 < \cdots < r_t < i$ and values $s_1, \ldots, s_m$ such that $t + m \leq i$ and such that we have specified the values $f(r_j) = r_{j+1}$ for all $j < t$ and we have specified the values $f(s_j) = 0$ for all $j \leq m$ and such that no other values of $f$ have been specified. In particular, the value of $f(r_t)$ has not been specified. Thus, after $i$ oracle queries, $\leq i$ values of $f$ have been specified ($t$ and $m$ vary with $i$, of course).

Suppose the $(i+1)$-st oracle query is for the value of $f(u)$. If $f(u)$ has already been specified, no action is taken and the computation of $M$ continues with the already specified valued. If $u \neq r_t$, then specify that $f(u) = 0$; this makes $u$ one of the $s_j$'s. Otherwise, if $u = r_t$, fix $f(u)$ to be equal to the first value $r_{t+1} > r_t$ for which the value of $f$ has not yet been specified.

At the end of $M$'s computation, $f$ has been defined consistently and so that conditions (1)-(3) are satisfied. Since $M$ runs for at most $|a|^c$ steps for some constant $c$, we take $a$ large enough so that $a > |a|^c$. Clearly $M$ can not reliably output a value $b$ such that $f(b) = a$; since, for any particular $b$ either $b$ is among $r_t$'s and then $f(b) < a$, or it is possible to set $f(b) = 0$ consistently with conditions (1)-(3).
Q.E.D. Theorem 4.4

(4.5) For technical reasons, we slightly generalize the iteration principle to a principle $Iter(f, a, a_0)$ by replacing conditions (1) and (2) by:

(1′) $a_0 < a \wedge a_0 < f(a_0)$.

(2′) $(\forall x < a)(a_0 \leq f(x) \rightarrow (f(x) = a \vee f(x) < f(f(x)))$.

Obviously, $Iter_0(f, a)$ is just $Iter(f, a, 0)$.

It is interesting to note that the iteration principle is a simplified form of a Skolemization of the induction axiom for $(\exists y \leq x)\alpha(x, y)$ (compare to Krajíček [9]). To see this, let the Skolemization of the induction axiom for $(\exists y \leq x)\alpha(x, y)$ be

$$(\alpha(0,0) \wedge \forall x, y \leq a\, ((\alpha(x,y) \wedge y \leq x) \rightarrow (\alpha(x+1, g(x,y)) \wedge g(x,y) \leq x+1)) \\ \rightarrow (\exists b \leq a)\alpha(a, b).$$

Consider the pairing function $[x, y] := x(a+1) + y$ and let $f$ be the function such that

$$f([x,y]) = \begin{cases} [x+1, g(x,y)] & \text{if } y \leq x < a \text{ and } \alpha(x,y) \\ (a+1)^2 & \text{if } x = a \text{ and } \alpha(x,y) \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that if the hypothesis of the Skolemization is satisfied, then $f$ satisfies the hypothesis of $Iter(f, (a+1)^2, 0)$ and thus $Iter(f, (a+1)^2, 0)$ implies that there is a pair $[x, y] < (a+1)^2$ such that $f([x, y]) = (a+1)^2$. From the definition of $f$, $x = a$ and $\alpha(a, y)$ and $y < a$, i.e., $(\exists b \leq a)\alpha(a, b)$ is true.

(4.6) A unary function $f : a \rightarrow a$ can be coded as a binary relation $\beta(x, i)$ on $a \times |a|$ by letting $\beta(x, i)$ be true if and only if the $i$-th bit of the binary representation of $f(x)$ is equal to 1. The predicate $\beta$ is called the *bit graph* of $f$. A formula $f(x) = y$ is then equivalent to the sharply bounded formula

$$y < a \wedge (\forall i < |a|)((y)_i = 1 \leftrightarrow \beta(x, i)),$$

where $(y)_i$ denotes the $i$-th bit of the binary representation of $y$. So by standard techniques, any $\Sigma_i^b$-formula $C(f)$ involving $f$ can be rewritten as an equivalent $\Sigma_i^b$-formula $C'(\beta)$ containing $\beta$ instead of $f$ (see Theorem 2.2 of [2]). Furthermore, w.l.o.g., every occurence of $\beta$ in $C'(\beta)$ has only bound variables as arguments. This allows us to generalize the concept of $A^{i,\alpha}$ from (2.5) to functions; namely, with $k = 2$, $A^{i,\alpha}(a, x_1, x_2)$ can be viewed as the bit graph of a function $F^{i,\alpha} : a \to a$ so that $F(x_1)$ has $x_2$-th bit equal to 1 iff $A^{i,\alpha}(a, x_1, x_2)$ holds.

This treatment of functions as relations also translates to oracle machines, namely, one oracle query about a function's value can be replaced by $|a|$ many queries about the bit graph of the function.

Let $Iter(F^{i,\alpha}, a, a_0)$ be the $\Sigma^b_{i+1}$-formula obtained by first expressing $Iter(f, a, a_0)$ as an equivalent $\Sigma^b_1$-formula involving the bit graph $\beta$ of $f$ instead of $f$, and then replacing every $\beta(y, z)$ by the formula $A^{i,\alpha}(a, y, z)$.

(4.7) **Theorem** *For $i \geq 0$, the $\Sigma^b_{i+1}(\beta)$-formula $Iter(F^{i,\beta}, a, a_0)$ is provable in $T^{i+1}_2(\beta)$ but not in $S^{i+1}_2(\beta)$.*

**Proof** The proof that $\Sigma^b_{i+1}$-IND implies the iteration principle is completely analogous to the proof of the first part of Theorem 4.4; we leave it to the reader to check the details.

It remains to show that $S^{i+1}_2(\beta)$ does not prove $Iter(F^{i,\beta}, a, a_0)$; assume, for the sake of a contradiction, that it does prove this. Then, by [2], $Iter(F^{i,\beta}, a, a_0)$ is $\Box^p_{i+1}$-witnessed, i.e., there is a polynomial time Turing machine $M$ with a $\Sigma^b_i(\beta)$-oracle that on inputs $a$ and $a_0$ produces a witness for $Iter(F^{i,\beta}, a, a_0)$. By the Collapsing Theorem 4.3 this implies that for many functions $f : a \to a$, $Iter(f, a, a_0)$ is "nearly" $\Box^p_1(f)$-witnessed. More precisely, there is a polynomial time machine $M^\beta$ and for any sufficiently large $m$ a $\Delta^{S,t}_1$-circuit oracle $C$ with variables from $B_{k,0}(m)$ so that $S = 2^{(\log m)^c}$ and $t = \log S$ for some constant $c$ ,and there is a set $Q \subseteq m \times \log(m)$ with the cylinder property (ii) of Theorem 4.3 holding, such that whenever $f : m \to m$ is coded by $\beta \subseteq m \times \log(m)$ with $\beta \cap Q = \emptyset$ and whenever $m_0 < m$, then the machine $M$ with circuit-oracle $C$ outputs a witness to $Iter(f, m, m_0)$. Since we will consider only functions $f$ which satisfy the hypotheses $1'$, $2'$ and 3 of the iteration principle, the witness output by $M$ will be a value $b$ such that $f(b) = m$.

We shall prove that no such machine $M$ with $\Delta^{S,t}_1$-circuit oracle $C$ exists; this suffices to show that $S^{i+1}_2$ does not prove $Iter(F^{i,\beta}, a, a_0)$.

Our stategy is to diagonalize against an execution of $M$ to produce a $\beta$ which codes a function $f$ satisfying the three hypotheses of the iteration principle but for which $M$ fails to output a value $b$ such that $f(b) = m$.

Each time an $M$ makes an oracle query we shall set sufficiently many values of $\beta$ so as to fix the answer to the query (no matter how $\beta$ is extended in the future). We shall adopt the convention that $\beta(x,j)$ will be false if $x \geq m$ or if $j > \log m$. We also adopt the convention that whenever a truth value of $\beta(x,j)$ is set (that is the value of the $j$-th bit of $f(x)$ is specified), then the rest of the the values $\beta(x,s)$, for $s \leq \log m$, are set (so that the value of $f(x)$ is completely specified). Thus, at any point during the construction of $\beta$, if $x < m$, then either $f(x)$ is completely unspecified or a value for $f(x)$ has been chosen.

We construct $\beta$ by executing $M$ with a fixed, sufficiently large $m$: after the $q$-th query of $M$ we shall have constructed a partial relation $\beta_q \subseteq m \times \log m$ which defines a partial function $f_q : m \to m$. (A *partial relation* is a partially specified relation in which some values of $\beta_q$ are set and others are yet undefined.) Initially, we let the domain $dom(f_0)$ of $f_0$ be the set of $x$ for which $\langle x,j\rangle$ is in $Q$, for some $j$ and set $f$'s value to be zero on its domain. And $\beta_0$ is the corresponding partial relation; namely, $\beta_0(x,s) = 0$ iff $\langle x,j\rangle \in Q$ for some $j \leq \log m$. We let $m_0$ be the least value not in $dom(f_0)$ and begin the execution of $M$ on the inputs $m$ and $m_0$.

For conceptual clarity, we shall transform the $\Delta_1^{S,t}$-circuits of the oracle circuits which use the function $f$ in place of the relation $\beta$. Each circuit $C_u^{\pm}$ consists of an OR of AND's, each of fanin $\leq t$ (recall that the family $C$ contains a pair of circuits $C_u^+$, $C_u^-$ for each possible oracle query $u$). The literals in the AND's are assertions of the form $\beta(x,s)$ or $\neg\beta(x,s)$. Each such literal may be replaced by an OR of the at most $m/2$ assertions $f(x) = y$ compatible with the assertion. After this replacement, the circuit may be put back into disjunctive normal form, yielding a circuit which consists of an OR of AND's, each of fanin $\leq t$ — now each input to an AND is an assertion of the form $f(x) = y$. Each AND may obviously be thought of as specifying a partial map with domain of size $\leq t$. For the rest of this proof, we shall consider the $C_u^{\pm}$'s as being in this form, as it makes our arguments easier to understand (this doesn't change the argument in any essential way).

After $M$'s $k$-th oracle query, we shall have defined a partial function $f_k \supseteq f_{k-1} \supseteq \cdots \supseteq f_0$ and a sequence $m_0 < m_1 < \cdots < m_{s_k}$ satisfying the following conditions:

(1) $|dom(f_k)| \leq |dom(f_0)| + kt^2 \leq \sqrt{m}\log m + k(\log m)^{2c}$.

(2) For $j < s_k$, $f_k(m_j) = m_{j+1}$; and $f_k(m_{s_k})$ is undefined.

(3) For all $v \in dom(f_k) \setminus \{m_0, \ldots, m_{s_k-1}\}$, $f_k(v) = 0$.

(4) $s_k \leq kt$ and $m_{s_k} \leq \sqrt{m} \log m + kt^2$.

(5) Any $f \supseteq f_k$ gives the same answers as $f_k$ to $M$'s first $k$ oracle queries.

These five conditions are clearly already fulfilled for $k = 0$ at the beginning of $M$'s execution (conditions (1) and (4) holds because the cylinder property (ii) of Theorem 4.3 is satisfied by $Q$.) We must ensure that these conditions remain true for the entire computation of $M$ — note that these conditions imply that $f_k$ can be extended (in many ways) to a total function satisfying the hypotheses of the iteration principle.

Now we describe how to define $f_{k+1}$ at $M$'s $(k+1)$-st oracle query. Suppose $M$'s $(k+1)$-st query is $u$, so the oracle answer is computed by the $\Delta_1^{S,t}$-circuit $C_u$ consisting of two $\Sigma_1^{S,t}$-circuits $C_u^+$ and $C_u^-$ computing each other's complements. We will define $f_{k+1}$ from $f_k$ adding at most $t^2$ elements to the domain so that one of $C_u^+$ and $C_u^-$ is forced to be true and so that conditions (1)-(5) hold.

The circuits $C_u^{\pm}$ each comprise an OR and AND's; each AND is a conjunction of $\leq t$ statements of the form $f(x) = y$. Thus each AND corresponds in the obvious way to a partial function $g$ with domain of cardinality $\leq t$ (namely, $g$ is the minimal partial function such that $f = g$ satisfies the AND). Let $pf(C_u^+)$, respectively, $pf(C_u^-)$ be the set of partial functions corresponding to the AND's of the circuits $C_u^+$, respectively, $C_u^-$. It is an elementary fact, that for any $g \in pf(C_u^+)$ and any $h \in pf(C_u^-)$ there must be a value $x$ such that that $g(x)$ and $h(x)$ are defined and are unequal; otherwise there would be a total function $f \supset g \cup h$ which would satisfy both $C_u^+$ and $C_u^-$.

If there is no $g \in pf(C_u^+)$ which is compatible with $f_k$ then $f_k$ already forces $C_u^-$ true and we set $f_{k+1} := f_k$. Otherwise, pick any $g_1 \in pf(C_u^+)$ which compatible with $f_k$ and choose $m_{s_k+1}$ to be least number greater than $m_{s_k}$ which is not in $dom(g_1) \cup dom(f_k)$. Let $k_1$ be the partial function with $dom(k_1)$ equal to $dom(f_k) \cup dom(g_1) \cup \{m_{s_k}\}$ and defined by

$$k_1(x) \quad = \quad \begin{cases} f_k(x) & \text{if } x \in dom(f_k) \\ m_{s_k+1} & \text{if } x = m_{s_k} \\ 0 & \text{if } x \in dom(g_1) \setminus dom(f_k) \text{ and } x \neq m_{s_k}. \end{cases}$$

Now if $k_1$ forces either $C_u^+$ or $C_u^-$ to be true, we set $f_{k+1} := k_1$. Otherwise, note that for each $h \in pf(C_u^-)$ there is at least one value in $dom(k_1) \cap dom(h)$; in other words, there are at most $t-1$ values in $dom(h) \setminus dom(k_1)$. Now pick an arbitrary $g_2 \in pf(C_u^+)$ which is compatible with $k_1$ and choose $m_{s_k+2}$ to be equal to the least value greater than $m_{s_k+1}$ not in $dom(g_2) \cup dom(k_1)$. Define the partial function $k_2$ from $k_1$, $g_2$ and $m_{s_k+2}$ in exactly the same fashion as $k_1$ was defined from $f_k$, $g_1$ and $m_{s_k+1}$. As before, either $k_2$ forces one of $C_u^+$ or $C_u^-$ to be true and we set $f_{k+1} := k_2$; or we have that for all $h \in pf(C_U^-)$, there are at most $t-2$ values in $dom(h) \setminus dom(k_2)$. We iterate this process until we find a $k_\ell$ with $\ell \leq t$ such that $k_\ell$ forces one of $C_u^+$ and $C_u^-$ to be true; then we set $f_{k+1} := k_\ell$. It is straightforward to verify that $f_{k+1}$ satisfies conditions (1)-(5).

The above completes the definition of the $f_k$'s. Since $M$ runs in polynomial time we choose $c$ so that $M(m)$ makes $k \leq (\log m)^c$ queries. $f_k$ is the partial function constructed at the end of the above process. By condition (4), we have

$$\begin{aligned} m_{s_k} &\leq \sqrt{m}\log m + k \cdot t^2 \\ &\leq \sqrt{m}\log m + (\log m)^c (\log m)^{2c} \\ &<< m \end{aligned}$$

for $m$ sufficiently large. Likewise $|dom(f_k)| << m$. Now $M$ cannot reliably output a witness to the iteration principle $Iter(f, m, m_0)$ since, for any output value $b$ of $M(m)$, we may extend $f_k$ to a total function $f$, such that $f$ satisfies the hyptheses $1'$, $2'$, 3 of the iteration principle and such that $f(b) \neq a$; namely, if $b \neq m_{s_k-1}$ let $f$ have value 0 whenever $f_k$ is undefined, and if $b = m_{s_k-1}$ let $f(m_{s_k}) = m_{s_k} + 1$ and otherwise have value 0 whenever $f_k$ is undefined.
Q.E.D. Theorem 4.7.

# 5 $T_2^1$ and Polynomial Local Search

(5.1) In [7] a *Polynomial Local Search* problem (PLS-problem) $L$ is defined to be a maximization problem satisfying the following conditions: (we have made some inessential simplifications to the definition in [7])

- For every instance $x \in \{0,1\}^*$, there is a set $F_L(x)$ of *solutions,* an integer valued cost function $c_L(s,x)$ and a neighborhood function $N_L(s,x)$,
- The binary predicate $s \in F_L(x)$ and the functions $c_L(s,x)$ and $N_L(s,x)$ are polynomial time computable. And there is a polynomial $p_L$ so that for all $s \in F_L(x)$, $|s| \leq p_L(|x|)$. Also, $0 \in F_L(x)$.
- For all $s \in \{0,1\}^*$, $N_L(s,x) \in F_L(x)$.
- For all $s \in F_L(x)$, if $N_L(s,x) \neq s$ then $c_L(s,x) < c_L(N_L(s,x),x)$.
- The problem is solved by finding a locally optimal $s \in F_L(x)$, i.e., an $s$ such that $N_L(s,x) = s$.

It follows from these conditions that there is a polynomial time computable $M_L(x)$ such that $M_L(x) > c_L(s,x)$ for all $s \in F_L(x)$.

A PLS-problem $L$ can be expressed as a $\Pi_1^b$-sentence saying that the conditions above hold; if these are provable in $T_2^1$ then we say $L$ is a PLS-problem in $T_2^1$. The formula $Opt_L(x,s)$ is the $\Delta_1^b$-formula $N_L(s,x) = s$.

(5.2) **Theorem** *Let $L$ be a PLS-problem in $T_2^1$. Then* $T_2^1 \vdash (\forall x)(\exists y)Opt_L(x,y)$.

**Proof** It is known [2] that $T_2^1$ proves the $\Sigma_1^b$-MIN axioms; this immediately implies also the $\Sigma_1^b$-MAX principle. Arguing informally in $T_2^1$, we have that, for all $x$, there is a maximum value $c_0 < M_L(x)$ satisfying $(\exists s \in F_L(x))(c_L(s,x) = c_0)$. Taking $s$ to be witness for this last formula, $s$ is globally optimal and hence satisfies $Opt_L(x,s)$, and the theorem is proved. Q.E.D. Theorem 5.2

(5.3) Now we establish a converse to Theorem 5.2. We shall use the definition of the formula *Witness* from [2]. We also adopt the convention that witnesses are efficiently coded, i.e., for every $\Sigma_1^b$-formula $C(\vec{u})$ there is a term $t_C(\vec{u})$ so that any witness for $C(\vec{u})$ must be $\leq t_C(\vec{u})$, as in Theorem 5.3 of [2].

**Theorem** *Let $\theta(a)$ be a $\Sigma_1^b$-formula such that $T_2^1 \vdash (\forall x)\theta(x)$. Then there is a PLS-problem $L$ in $T_2^1$ such that $T_2^1$ proves*

$$(\forall x)(\forall s)(Opt_L(x,s) \rightarrow Witness_\theta^{1,a}(s,x)).$$

The point of the previous two theorems is that, on one hand, any PLS-problem can be expressed as a $\Sigma_1^b$-defined function in $T_2^1$ and that, conversely, any $\Sigma_1^b$-function of $T_2^1$ can be expressed as a PLS-problem composed with a projection function.

**Proof** If $T_2^1$ proves $(\forall x)\theta(x)$, then by free-cut elimination, there is a $T_2^1$-proof $P$ in the Gentzen sequent calculus system LKB of the sequent $\longrightarrow\theta(u_1)$ such that every sequent in $P$ is of the form

$$A_1(\vec{u}), \ldots, A_k(\vec{u}) \longrightarrow B_1(\vec{u}), \ldots, B_\ell(\vec{u})$$

where $\vec{u}$ is a vector of $r$ free variables (which includes the variable $u_1$) and where all the formulas $A_i$ and $B_i$ are $\Sigma_1^b$-formulas.

We shall prove by induction on the number of proof steps that any sequent of the above form provable in $T_2^1$ corresponds computationally to a PLS-problem. Namely, there is a PLS-problem $L'$ such that (1) inputs to $L'$ are (encodings of) $k+r$-tuples $\langle m_1, \ldots, m_r, v_1, \ldots, v_k\rangle$ where $m_1, \ldots, m_r$ are values for the variables $u_1, \ldots, u_r$ and (2) for input a tuple $\langle \vec{m}, \vec{v}\rangle$, the locally optimal solutions are the $k+r+1$-tuples of the form $\langle \vec{m}, \vec{v}, w\rangle$ with the same $\vec{m}$ and $\vec{v}$ values such that if each $v_i$ witnesses $A_i(\vec{m})$ then $w$ is a witness for one of the formulas $B_1(\vec{m}), \ldots, B_\ell(\vec{m})$. From such problem $L'$ we get problem $L$ satisfying the requirement of the theorem by adding to each $L'$-solution $\langle \vec{v}, w\rangle$ a new neighbour $w$ with higher cost, provided $w$ is a witness to $\theta$.

The existence of the PLS-problem is obvious for initial sequents, which by definition contain only atomic formulas. The induction step splits into cases depending on the final inference of the proof $P$. The cases where the final inference is a propositional inference or a structural inference other than cut are very simple, requiring only minor changes to the PLS-problem. The case where the final inference of $P$ is an $\exists$*:right* inference

$$\frac{? \longrightarrow \Delta, A(t)}{t \leq s, ? \longrightarrow \Delta, (\exists x \leq s)A(x)}$$

can be handled easily also: the induction hypothesis states that there is a PLS problem $L$ that applies to the upper sequent. We now sketch how to modify $L$ to construct a PLS problem $L'$ that works for the lower sequent. First, let $c_{L'}(s,x) = c_L(s,x) + 1$ for $s \in F_L(x)$. Inputs $\langle \vec{m}, v_0, \vec{v}\rangle$ to $L'$ that provide witnesses to ? are assigned cost 0 and have as neighbour the input $\langle \vec{m}, \vec{v} \rangle$ to $L$. An output $\langle \vec{m}, \vec{v}, w \rangle$ of $L$ has as its $L'$-neighbour a tuple $\langle \vec{m}, v_0, \vec{v}, w' \rangle$ with cost $M_L(\langle \vec{m}, \vec{v} \rangle) + 1$, where $w' = w$ or $w' = \langle t(\vec{m}), w \rangle$, whichever provides a witness to a formula in the succedent $\Delta, (\exists x \leq s)A$. It is easily checked that $L'$ has the desired properties.

Similarly the case where the final inference of $P$ is an $\exists \leq$*:left* or a $\forall$*:left* is handled by simple modifications to the PLS-problem. The case where the final inference is a $\forall$*:right* is more complicated: it is comparable to the case where the final inference is an induction rule (treated below) and we leave it to the reader.

In the case where the final inference of $P$ is a cut inference

$$\frac{\Pi \longrightarrow A \qquad A \longrightarrow \Delta}{\Pi \longrightarrow \Delta}$$

we have, by the induction hypothesis, two PLS-problems $L_1$ and $L_2$ which apply to the upper sequent. A PLS problem for the lower sequent is formed as a "composition" of PLS problems. (To simplify this case, we assume w.l.o.g. that the cut formula $A$ is the only formula in the succedent (antecedent) of the left (right, resp.) upper sequent.) By coding, the PLS problems $L_1$ and $L_2$ can be modified to have domains $F_{L_1}$ and $F_{L_2}$ disjoint. The local optima (outputs) of the PLS problem $L_1$ can have as neighbours inputs to $L_2$. By adding $M_{L_1}(\cdots)$ to the cost function of $L_2$, the cost of any $L_2$-solution is greater than the cost of any $L_1$-solution. This makes it possible to arrange that any local optimum of the PLS combined problem can be found by applying $L_2$ to a local optimum of $L_1$. We leave the precise details to the reader.

Finally consider the case where the final inference of $P$ is an induction inference

$$\frac{A(u_0, \vec{u}) \longrightarrow A(u_0 + 1, \vec{u})}{A(0, \vec{u}) \longrightarrow A(t(\vec{u}), \vec{u})}$$

W.l.o.g., there are no side formulas to the induction inference.* Given a PLS problem $L$ for the upper sequent, we form a PLS-problem $L'$ for the lower sequent. The general idea is, of course, that $L'$ is an exponentially long iteration of instances of $L$. First, the set $F_{L'}(\langle \vec{m}, v \rangle)$ is the set of tuples

*This is because we may conjoin and disjoin any side formulas, which must be $\Sigma_1^b$-formulas, into the induction formula. This modification uses only propositional inferences.

$\langle m_0, z, s\rangle$ where $m_0 < t(\vec{m})$ and $s \in F_L(\langle m_0, \vec{m}, z\rangle)$; thus $F_{L'}$ is a disjoint union of solution spaces for instances of $L$. We define

$$c_{L'}(\langle m_0, z, s\rangle, \langle \vec{m}, v\rangle) \;=\; m_0 \cdot M + c_L(s, \langle m_0, \vec{m}, z\rangle)$$

where $M$ is a function of $\langle m, \vec{z}\rangle$ and is large enough to dominate $M_L(s)$ whenever $m_0 < t(\vec{m})$ and $s \in F_L(\langle m_0, m, z\rangle)$. The neighbourhood function is defined so that

$$N_{L'}(\langle m_0, z, s\rangle, \langle \vec{m}, v\rangle) \;=\; \langle m_0, z, N_L(s, \langle m_0, \vec{m}, z\rangle)\rangle$$

except when $s = N_L(s, \langle m_0, \vec{m}, z\rangle)$, in which case, for $m_0 < t(\vec{m}) - 1$, we set

$$N_{L'}(\langle m_0, z, s\rangle, \langle \vec{m}, v\rangle) \;=\; \langle m_0 + 1, z', \langle m_0 + 1, \vec{m}, z'\rangle\rangle$$

where $z'$ is the last component of $s$, i.e., the witness for $A(m_0 + 1, \vec{m})$. When $m_0 = t(\vec{m}) - 1$, then

$$N_{L'}(\langle m_0, z, s\rangle, \langle \vec{m}, v\rangle) \;=\; \langle \vec{m}, v, z'\rangle.$$

This last case gives a local optimum for $L'$. It is easy to check (and we leave it to the reader) that $L'$ gives a PLS problem that solves the lower sequent of the induction inference.
Q.E.D. Theorem 5.3

(5.4) There are two open problems concerning PLS problems and $T^1_2$ that are interrelated by Theorems 5.2 and 5.3. First, can any PLS problem $L$ be PLS reduced in the sense of [7] to a PLS-problem which has, for all inputs, a unique local optimum? And second, is it true that whenever $T^1_2$ proves $(\exists x \le t)A$ with $A \in \Sigma^b_1$ then there exists a $\Sigma^b_1$-formula $B$ such that $T^1_2$ proves $(\exists! x \le t)B$ and $B \to A$? These questions are not apparently equivalent since even if local optima are unique, they may not be provably unique in $T^1_2$.

Papadimitriou [13] has introduced two classes $PLF$ and $PLDF$ of search problems and showed that $PLDF \subseteq PLF$.† A $PLDF$ search problem $L$ has, for every input $x$ a directed graph $N_x(c, c')$ on nodes $c, c' < t(x)$ for some

†In a later paper [14], the classes $PLF$ and $PLDF$ are renamed to $PPA$ and $PPAD$, respectively.

term of Bounded Arithmetic, such that every node has indegree and outdegree $\leq 1$. In addition, it is assumed that $N_x(c, c')$ is a polynomial time predicate of $x$, $c$, and $c'$ and that if there exists a value $c'$ (resp., $c$) such that $N_x(c, c')$ holds, then it can be computed in polynomial time from $x$ and $c$ (resp., from $x$ and $c'$). On input a pair $\langle x, c_0 \rangle$ such that $c_0$ has indegree 0 in $N_x$, the problem is to find a node that has outdegree 0: such a node must exist since the directed graph is finite. However, it is unlikely that $T^1_2$ can prove that PLDF problems must have solutions since the pigeon-hole principle can be reduced to the statement that a PLDF problem has a solution. For instance, if $f$ and $g$ are new function symbols, we can define a graph $N(c, c')$ by the condition $f(c) = c'$ and $g(c') = c$. Now if $g$ is further presumed to be the inverse of $f$ then the pigeonhole principle for $f$ is equivalent to the statement that if $N$ has a node of indegree 0 then $N$ must have a node of outdegree 0. However, by [16, 11, 1], the pigeonhole principle for $f$ is not provable even in $T_2(f)$. Thus $T_2(f, g)$ does not prove the existence of solutions for this PDLF problem.

# References


[1] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák and A. Woods, *Exponential lower bounds for the pigeonhole principle*, to appear in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992.

[2] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[3] ——, *Axiomatizations and conservation results for fragments of bounded arithmetic*, in Logic and Computation, proceedings of a Workshop held Carnegie-Mellon University, 1987, vol. 106 of Contemporary Mathematics, American Mathematical Society, 1990, pp. 57–84.

[4] M. Furst, J. B. Saxe, and M. Sipser, *Parity, circuits and the polynomial-time hierarchy*, Math. Systems Theory, 17 (1984), pp. 13–27.

[5] P. Hájek and P. Pudlák, *Metamathematics of First-order Arithmetic*, Springer-Verlag. To appear.

[6] J. HASTAD, *Almost Optimal Lower Bounds for Small Depth Circuits*, in: Randomness and Computation, ed. S.Micali, Ser.Adv.Comp.Res., vol. 5, JAI Press, 1989, pp. 143–170.

[7] D. S. JOHNSON, C. H. PAPADIMITRIOU, AND M. YANNAKAKIS, *How easy is local search?*, J. Comput. System Sci., 37 (1988), pp. 79–100.

[8] J. KRAJÍČEK, *Fragments of bounded arithmetic and bounded query classes*. To appear in Transactions of the A.M.S.

[9] ——, *No counter-example interpretation and interactive computation*, in Logic From Computer Science: Proceedings of a Workshop held November 13-17, 1989, ed. Y.N.Moschovakis, Mathematical Sciences Research Institute Publication #21, Springer-Verlag, 1992, pp. 287–293.

[10] J. KRAJÍČEK, P. PUDLÁK, AND G. TAKEUTI, *Bounded arithmetic and the polynomial hierarchy*, Annals of Pure and Applied Logic, 52 (1991), pp. 143–153.

[11] J. KRAJÍČEK, P. PUDLÁK, AND A. WOODS, *Exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle*, submitted, (1991).

[12] J. KRAJÍČEK AND G. TAKEUTI, *On induction-free provability*, Annals of Mathematics and Artificial Inteligence,6, (1992), pp.107-126.

[13] C. H. PAPADIMITRIOU, *On graph-theoretic lemmata and complexity classes (extended abstract)*, in Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (Volume II), IEEE Computer Society, 1990, pp. 794–801.

[14] C. H. PAPADIMITRIOU, *On the complexity of the parity argument and other inefficient proofs of existence*, to appear in J. Comput. System Sci.

[15] J. B. PARIS, A. J. WILKIE, AND A. R. WOODS, *Provability of the pigeonhole principle and the existence of infinitely many primes*, Journal of Symbolic Logic, 53 (1988), pp. 1235–1244.

[16] T. PITASSI, P. BEAME, AND R. IMPAGLIAZZO, *Exponential lower bounds for the pigeonhole principle*, preprint, (1991).

[17] P. Pudlák, *Some relations between subsystems of arithmetic and the complexity of computations*, in Logic From Computer Science: Proceedings of a Workshop held November 13-17, 1989, ed.Y.N.Moschovakis, Mathematical Sciences Research Institute Publication #21, Springer-Verlag, 1992, pp. 499–519.