Electronic Colloquium on Computational Complexity, Report No. 51 (2003)

The Relative Complexity of Local Search Heuristics and the Iteration Principle

Tsuyoshi Morioka
(morioka@cs.toronto.edu)
Department of Computer Science
University of Toronto

June 20, 2003

Abstract. Johnson, Papadimitriou and Yannakakis introduce the class **PLS** consisting of optimization problems for which efficient local-search heuristics exist. We formulate a type-2 problem **ITER** that characterizes **PLS** in style of Beame et al., and prove a criterion for type-2 problems to be nonreducible to **ITER**. As a corollary, we obtain the first relative separation of **PLS** from Papadimitriou's classes **PPA**, **PPAD**, **PPADS**, and **PPP**. Based on the criterion, we derive a special case of Riis's *independence criterion* for the Bounded Arithmetic theory $S_2^2(L)$. We also prove that **PLS** is closed under Turing reducibility.

1 Introduction

Complexity theory has been successful in classifying various computational problems in terms of the required amount of resources for solving them. In particular, a vast number of combinatorial optimization problems that arise in practical applications have been shown to be NP-hard.

One successful approach to such difficult optimization problems is *local search*. Given an optimization problem, one specifies a local-search heuristic and then tries to find a locally-optimal solution, or a solution whose quality cannot be improved by the heuristic.

In order to study the complexity of local search, Johnson, Papadimitriou, and Yannakakis [JPY88] introduce the class Polynomial Local Search (**PLS**) consisting essentially of optimization problems for which polynomial-time local-search heuristics exist. **PLS** contains many natural problems, some of which are complete for the class (with respect to an appropriate notion of reduction). Unlike many standard complexity classes such as **P** or **NP**, which are classes of decision problems, **PLS** is a class of *total search problems*.

Beame et al. [BCE+98] introduce a framework to study the complexity of total search problems. Their framework is based on the notion of type-2 computation of [Tow90, CIY97], whose inputs consist of not only strings (type-1 objects) but also functions and relations (type-2 objects).

The type-2 approach is useful in the study of 'algorithmic power' of combinatorial principles since witnessing a combinatorial principle in a finite structure can be nicely formulated as a total type-2 search problem. Various type-2 problems are introduced by Beame et al. from the combinatorial principles such as the pigeonhole principle. These type-2 problems give rise to alternative type-2 definitions for Papadimitriou's search classes **PPA**, **PPAD**, **PPADS**, and **PPP** of [Pap94b], which contain a number of natural combinatorial problems.

All possible containments and relative separations among these classes are proven in [BCE+98] by showing reducibilities and nonreducibilities among the corresponding search problems. However, **PLS** is not discussed by Beame et al., and it has been unknown how **PLS**

ISSN 1433-8092

fits in with respect to the other search classes in a relativized world. This paper solves this open problem.

In his Ph.D thesis [Bus86], Buss introduces a hierarchy of fragments S_2^i and T_2^i of Peano Arithmetic called Bounded Arithmetic and shows its close connection to the Polynomial-time Hierarchy via the notions of *definable functions* and *witness theorems*. Buss and Krajicek [BK94] obtain a logical characterization of **PLS** in Bounded Arithmetic, which can be stated as follows: a problem is in **PLS** if and only if it is Σ_1^b-definable in T_2^1. It is interesting that **PLS**, which arose in an attempt to capture local-search heuristics for practical applications, coincides with a definable class of search problems in a fragment of Peano Arithmetic. Subsequently Chiari and Krajicek [CK98] show that **PLS** is characterized (in a precise sense) by a combinatorial principle called the *iteration principle*, which is a variant of the induction principle.

In this paper we introduce a new type-2 search problem **ITER** based on the iteration principle, similarly to the way Chiari and Krajicek define the *I-problem* [CK98]. **ITER** has the following property: for a search problem Q, $Q \in$ **PLS** if and only if Q is many-one reducible to **ITER**.

We then present a result regarding the power of Turing reducibility and many-one reducibility with respect to **ITER**.

Theorem 7 *Let Q be a search problem (type-1 or type-2). If Q is Turing reducible to* **ITER**, *then Q is many-one reducible to* **ITER**.

It follows that **PLS** is closed under Turing reducibility, which, as far as we know, has not been known.

We introduce the following, systematic method of formulating type-2 search problems: given an existential first-order sentence Φ, the corresponding type-2 search problem Q_Φ is the problem of finding a witness to Φ in a given finite structure. If the underlying first-order language contains $\leq$, then the input structure interprets $\leq$ as the standard linear ordering.

The main result of this paper is a sufficient condition on Φ for Q_Φ to be nonreducible to **ITER**.

Main Theorem *Let Φ be a first-order existential sentence over a language L not containing $\leq$. If Φ fails in an infinite structure, then the corresponding type-2 search problem Q_Φ is not Turing reducible to* **ITER**.

For example, Main Theorem applies to the following sentence:

$$(\forall x)[\alpha(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge \alpha(x) = \alpha(y)].$$

This is the injective pigeonhole principle, which we will use as a primary example throughout this paper. It follows from Main Theorem that the above sentence cannot be witnessed in polynomial-time by making queries to **ITER**.

Since the combinatorial principles corresponding to Papadimitriou's classes can be stated in the form satisfying the premise of Main Theorem, we obtain the the first relative separation of **PLS** from Papadimitriou's classes.

Corollary 9 *There exists a relativized world in which* **PLS** *contains none of the following:* **PPA**, **PPAD**, **PPADS**, *and* **PPP**.

Corollary 9 may be interpreted as evidence that efficient local search heuristics are unlikely to exist for the problems in Papadimitriou's classes, which include problems related to cryptographic hash functions (**PPP**), finding Nash equilibria (**PPAD**), Sperner's lemma, Brouwer's fixed point theorem, and the Borsuk-Ulam theorem (**PPA**) [Pap94b].

Note that we have proven Corollary 9 directly in [Mor01]. Main Theorem of this paper is a generalization of the proof in [Mor01].

We will strengthen Main Theorem to hold for sentences of any quantifier complexity (Theorem 10).

Main Theorem is in the style of Krajicek's Theorem 11.3.1 in [Kra95]. We will discuss similarities and differences of the two statements in Section 5. Riis [Rii01] proves that the assumptions of Main Theorem implies that Φ gives rise

to tautologies that are hard for the tree-like resolution proof system. We do not know how his result relates to our Main Theorem.

Main Theorem implies the following 'independence criterion' for the relativized theory $T_2^1(L)$ of Bounded Arithmetic.

Theorem 16 *Let Φ be an $\exists$-sentence over a language L such that $L - \{0\}$ is disjoint with the language of Bounded Arithmetic. If Φ fails in an infinite structure, then*

$$T_2^1(L) \nvdash \Phi^{\leq a},$$

where $\Phi^{\leq a}$ is the $\Sigma_1^b(L)$-formula obtained by bounding all quantifiers with $2^{|a|}$.

Theorem 16 is a special case of Riis's independence criterion for $S_2^2(L)$ which applies to any first-order sentence [Rii93, Kra95]. Hence Theorem 16 itself does not give any new independence results for $T_2^1(L)$. However, some of the known independence results do follow from it.

Throughout this paper we work with binary strings. For string x, $|x|$ denotes the length of x. For any $n \geq 1$, V_n denotes the set $\{0,1\}^n$ of the strings of length n. The symbol '$\leq$' denotes the lexicographical ordering of strings. We write 0^n and 1^n to denote the sequences of n 0's and n 1's, respectively. When x and y are strings, $x \circ y$ denotes the concatenation of the two.

2 Search Problems

Let R be any binary relation. The *search problem* Q_R is the following problem: given x, find y such that $R(x,y)$. The input x is called an *instance* of Q_R and any y satisfying $R(x,y)$ is called a *solution* for instance x. For every x, $Q_R(x) = \{y : R(x,y)\}$ denotes the set of solutions for instance x. Usually we omit the subscript R. We say that Q is *total* if $Q(x)$ is nonempty for all x.

When a defining relation R is polynomial-time decidable, the resulting search problem is called an NP-search problem [BCE+98]. **TFNP** is defined to be the class of total NP search problems in [MP91] (see also [Pap94a]); the same class is called **VP** (for Verification of solutions in Polytime) in [Mor01].

There is some evidence in [BCE+98] that some total search problems may have no computationally equivalent decision problems. Hence, the complexity of total search problems, such as PLS problems below, needs to be investigated directly.

This paper is concerned with total search problems, and hence all problems we mention are assumed to be total.

We say that a Turing machine M *solves* a search problem Q if, for all x, M on x outputs some $y \in Q(x)$. We write **FP** to denote the class of all search problems solvable by a deterministic polynomial-time Turing machine. Note that our **FP** contains all polynomial-time functions.

2.1 Reducibility

A search problem Q is used as an oracle in the following way. An oracle Turing machine M writes a query q on its query tape and enters in a special query state. In the next step, some $a \in Q(q)$ magically appears in the answer tape.

Let Q_1 and Q_2 be two search problems. Then Q_1 is *Turing reducible to* Q_2, written $Q_1 \leq_T Q_2$, iff there exists a polynomial-time oracle Turing machine that solves Q_1 by using Q_2 as an oracle. We say that Q_1 is *many-one reducible to* Q_2 and write $Q_1 \leq_m Q_2$ iff Q_1 is Truing reducible to Q_2 by a machine that asks at most one query to Q_2.

2.2 Polynomial Local Search

Johnson, Papadimitriou, and Yannakakis [JPY88] defined a *polynomial local search* (PLS) problem as an optimization problem that can be formulated as a local search problem. The following is a simplified definition of PLS problems.

Definition 1 *A search problem Q is said to be a PLS problem if the following conditions are met:*

(1) there exists a polynomial p such that $Q(x) \subseteq V_{p(|x|)}$ for all x.

(2) there exist polytime functions q_Q and h_Q such that $h_Q(x, 0^n) \neq 0^n$ and, for all $v \in V_{p(|x|)}$,

if $v \neq h_Q(x,v)$ *then* $q_Q(x,v) < q_Q(x,h_Q(v))$.

(3) the solutions are defined as

$$Q(x) = \{v \in V_{p(|x|)} : v = h_Q(x,v)\}.$$

The set $V_{p(|x|)}$ is the search space in which we look for solutions. The function q_Q defines the quality of each potential solution, and h_Q is a local-search heuristic. The solutions for instance x are the locally optimal elements of $V_{p(|x|)}$, i.e., the elements for which the heuristic h_Q does not return an object with better quality. A large number of common local search heuristics give rise to PLS problems [JPY88, Yan97].

The class of all PLS problems is not closed under $\leq_m$ unless all PLS problems are polytime solvable [Mor01]. Hence, we artificially close the class under $\leq_m$, in the style of [BCE+98].

Definition 2 **PLS** *is the smallest class containing all PLS problems closed under many-one reducibility.*

Note that our **PLS** is apparently larger than the class with the same name in [JPY88, Yan97]; in particular, our **PLS** is not a subclass of **TFNP** unless **PLS** $\subseteq$ **FP** [Mor01]. However, many properties of the class of PLS problems are also true for **PLS**. For example, all PLS-complete problems of [JPY88, Yan97] are also complete for our **PLS**.

3 Combinatorial Principles and Search Problems

Beame et al. [BCE+98] demonstrate that it is useful to generalize the notion of search problem so that the instances of search problem Q consist of strings, which are type-1 objects, and functions, which are type-2 objects. More formally, let R be a type-2 relation with arguments (α, x, y), where x and y are strings and α is a string function. R defines a type-2 search problem Q_R in the usual way.

The complexity of type-2 relation, functions, and search problems is measured with respect to a Turing machine that receives the type-1 arguments on its input tape and is allowed to access the type-2 arguments as oracles. In particular, a type-2 function $F(\alpha, x)$ is said to be polynomial-time computable if it is computed by a deterministic Turing machine in time polynomial in $|x|$ with oracle access to α.

Let Q be a type-2 search problem. An oracle Turing machine M uses Q as an oracle in the following way. M presents a query to Q in the form (β, y), where y is a number and β encodes a polynomial-time function. In the next step M receives in its answer tape some z that is a solution for $Q(\beta, y)$.

Let Q_1 and Q_2 be type-2 search problems. We say Q_1 is *Turing reducible to* Q_2 and write $Q_1 \leq_T Q_2$ iff there exists an oracle Turing machine M that, given an instance (α, x) of Q_1, outputs some $z \in Q_1(\alpha, x)$ in polynomial-time using α and Q_2 as oracles, where each query to Q_2 is of the form (y, β) with β a function polynomial-time computable using α as an oracle.

Q_1 is *many-one reducible to* Q_2 if $Q_1 \leq_T Q_2$ by an oracle Turing machine that asks at most one query to Q_2.

The following definition is from [BCE+98] with a small modification.

Definition 3 *Let Q be a type-2 search problem. Then $C(Q)$ is defined as*

$$C(Q) = \{Q' : Q' \text{ is type-1 and } Q' \leq_m Q\}.$$

3.1 Combinatorial Principles as First-Order Sentences

We introduce below a new, systematic way of defining type-2 search problems from combinatorial principles that are representable by sentences of first-order logic with equality.

Let L be an arbitrary first-order language and let Φ be a sentence over L of the form

$$\Phi =_{syn} (\exists x_1 \ldots \exists x_k)\phi(x_1, \ldots, x_k)$$

for some quantifier-free ϕ. Let us call such sentences $\exists$-sentences. Note that we allow the equality symbol $=$ in Φ even though we do not explicitly include it in L.

Φ is interpreted in a structure $\mathcal{M}$ which defines the universe of discourse and the meaning of constants, functions, and relations of L. We define a *canonical structure* to be a structure such that (1) the universe of discourse is V_n for some $n \geq 1$; (2) if $0 \in L$, then 0 means $0^n \in V_n$; and (3) if $\leq \in L$, then $\leq$ is the lexicographic ordering of strings. We abuse the notation and write V_n to denote the canonical structure with the set V_n the universe of discourse.

Assume that Φ holds in every canonical structure. Then the corresponding witness problem is the following: given a canonical structure V_n, find a tuple $\langle v_1, \ldots, v_k \rangle \in (V_n)^k$ such that $\phi(v_1, \ldots, v_k)$ holds in V_n.

We formulate the witness problem as the type-2 search problem Q_Φ whose type-1 argument x specifies the universe of discourse $V_{|x|}$ and whose type-2 arguments are the functions and relations of L. We exclude $\leq$ from the inputs, since its meaning in V_n is already fixed. Finally, since only the length of x is used to define $V_{|x|}$, we assume without loss of generality that the type-1 argument of Q_Φ is always of the form 1^n for $n \geq 1$.

To simplify our presentation, we say that a sentence is $\exists$ if its prenex form is purely existential. For example, the following sentence, Φ_{PIGEON}, is a $\exists$-sentence:

$$(\forall x)[\alpha(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge \alpha(x) = \alpha(y)].$$

In V_n, it states that, if 0^n is not in the range of α, then α cannot be one-to-one. By the (injective) pigeonhole principle, this holds in every canonical structure (in fact, in every finite structure). The corresponding type-2 search problem, called **PIGEON**, is the following: given $(1^n, \alpha)$, find either $v \in V_n$ such that $\alpha(v) = 0^n$ or $u, v \in V_n$ such that $u \neq v$ and $\alpha(u) = \alpha(v)$. **PIGEON** is defined in [BCE+98], and we will discuss it more in Section 5.

3.2 The Iteration Problem and PLS

The *iteration principle* is a weak form of the induction principle. It can be stated as follows.

$$\alpha(0) > 0 \wedge (\forall x)[\alpha(x) \geq x] \supset (\exists x)[x < \alpha(x) \wedge \alpha(x) = \alpha(\alpha(x))],$$

where '$x < y$' is an abbreviation for '$x \leq y \wedge x \neq y$'. Note that the iteration principle holds in every canonical structure but fails in a finite noncanonical structure.

We define **ITER** to be the type-2 problem defined by the above sentence. That is, given an instance $(1^n, \alpha)$, each $v \in V_n$ is a solution for **ITER**$(1^n, \alpha)$ if and only if one of the following holds: $\alpha(0^n) = 0^n$; $\alpha(v) < v$; or both $v < \alpha(v)$ and $\alpha(v) = \alpha(\alpha(v))$.

The iteration principle is studied in the context of Bounded Arithmetic by Buss and Krajicek [BK94] and Chiari and Krajicek [CK98]. Buss and Krajicek essentially prove that **ITER** is not solvable in type-2 polynomial-time. They also give a remarkable characterization of **PLS** in Bounded Arithmetic, which we discuss in Section 6. Based on this result Chiari and Krajicek prove a close connection between the iteration principle and **PLS**.

The following statement, which is motivated by a result of Chiari and Krajicek [CK98], is explicitly stated and proven in [Mor01].

Theorem 4 **PLS** $= C($**ITER**$)$.

That is, a type-1 search problem Q is in **PLS** if and only if $Q \leq_m$ **ITER**. Thus, **PLS** can be defined as the class of all type-1 search problems that are many-one reducible to **ITER**.

The following is a variant of the theorem of [CIY97] (also in [BCE+98]).

Theorem 5 *Let Q_1 and Q_2 be two type-2 search problems. Then $Q_1 \leq_m Q_2$ if and only if, for all oracle A, $(C(Q_1))^A \subseteq (C(Q_2))^A$.*

Thus, by proving $Q \not\leq_m$ **ITER**, one can show that $(C(Q))^A \not\subseteq$ **PLS**^{A} for some oracle A. In Section 5 we discuss Main Theorem, which provides a sufficient condition for Q to be nonreducible to **ITER**.

3.3 Equivalent Problems

ITER is not the only type-2 problem that characterizes **PLS**, and we present another type-2 problem **BTROOT** (Binary Tree Root) that is equivalent to **ITER** in the sense that **BTROOT** and **ITER** are many-one reducible to each other. It follows that **BTROOT** characterizes **PLS**.

Let us first restate **ITER** in graph-theoretic terms. Given an instance $(1^n, \alpha)$, consider the directed graph G whose vertex set is V_n and (u, v) is a directed edge iff $u \neq v$ and $\alpha(u) = v$. It follows that G is a directed acyclic graph (dag) with outdegree ≤ 1. We define a *sink* of G to be a vertex with no outgoing edge and at least one incoming edge. Then **ITER** is essentially the problem of finding a sink in this exponentially large graph, and because of the principle "every dag with at least one edge has a sink", **ITER** is total. Note that the only upper bound on the indegree of G is exponential in n. Moreover, the oracle α only tells us the outgoing edge of each vertex, and it is impossible in general to find an incoming edge of the given vertex in polynomial time.

We define the type-2 problem **BTROOT** as the problem of finding sinks in an exponentially large dag with outdegree ≤ 1 and indegree ≤ 2 such that it is easy to find the incoming edges of a vertex. The underlying graph is a collection of binary trees and isolated nodes, and the task is to find the root of any tree, and hence the name **BTROOT**.

Instances of **BTROOT** are of the form $(1^n, \alpha, \beta_1, \beta_2)$, where α is intended to define the outgoing edge of a vertex and β_1 and β_2 give two incoming edges. **BTROOT** is defined by the following $\exists$-sentence over $L = \{0, \alpha, \beta_1, \beta_2, \leq\}$:

$$\begin{aligned}
&\alpha(0) > 0 \wedge (\forall x)[\alpha(x) \geq x] \wedge \\
&(\forall x)[\beta_1(x) \leq x \wedge \beta_2(x) \leq x] \wedge \\
&(\forall x \forall y)[x = y \vee \\
&\qquad ((\beta_1(y) = x \vee \beta_2(y) = x) \leftrightarrow \alpha(x) = y)] \\
&\supset (\exists x)[x < \alpha(x) \wedge \alpha(x) = \alpha(\alpha(x))].
\end{aligned}$$

Theorem 6 **ITER** *and* **BTROOT** *are many-one reducible to each other.*

A proof of Theorem 6 is in Appendix A.

We formulate another problem equivalent to **ITER** in a joint work with Buresh-Oppenheim [BOM03].

4 Closure Property of PLS

Note that, in [BCE+98], all known reductions between type-2 problems are many-one reductions and all known separations are with respect to Turing reduction. That is, we do not know whether any pair Q_1 and Q_2 exists such that Q_1 is Turing reducible to Q_2 but not many-one reducible; in other words, we do not know if Turing reduction is more powerful than many-one reduction. However, we prove that this is not true for **ITER**.

Theorem 7 *Let Q be a search problem (type-1 or type-2). $Q \leq_m$* **ITER** *if and only if $Q \leq_T$* **ITER**.

By Theorem 4, it follows that **PLS** is closed under Turing reducibility. As far as we know, this has not been known.

Theorem 7 simplifies the proof of Main Theorem, since, in order to prove $Q \not\leq_T$ **ITER**, we only need to show $\not\leq_m$. However, it is not hard to directly extend the proof of $\not\leq_m$ to $\not\leq_T$ without invoking Theorem 7; see [BCE+98].

5 Main Result

Main Theorem *Let Φ be an $\exists$-sentence over a first-order language L without the standard ordering $\leq$. If Φ fails in an infinite structure, then*

$$Q_\Phi \not\leq_T \textbf{ITER}.$$

In [CK98], Chiari and Krajicek essentially prove that, for the weak pigeonhole principle and the generalized iteration principle, the corresponding type-2 search problems are not many-one reducible to **ITER**. Their results follow as special cases of our Main Theorem. We also

prove a special case of Main Theorem in [Mor01] which implies Corollary 9.

Before proving Main Theorem in Section 5.3, let us illustrate what Main Theorem is about. Recall Φ_{PIGEON} from Section 3

$$(\forall x)[\alpha(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge \alpha(x) = \alpha(y)],$$

which defines the type-2 problem **PIGEON**. Since Φ_{PIGEON} fails in an infinite structure, it follows from Main Theorem that **PIGEON** is not Turing reducible to **ITER**.

It is important that L does not contain $\leq$. For example, **ITER** and **BTROOT**, which are many-one reducible to **ITER**, are defined by sentences containing the symbol $\leq$.

Main Theorem is similar to Theorem 11.3.1 of [Kra95] below. A $\exists\forall$-sentence is a sentence of the form $(\exists\overline{x})(\forall\overline{y})\phi$ with ϕ quantifier-free. A language is *relational* if it contains no function symbol. Let $(\mathbf{FP^{NP}})^2$ denote the class of type-2 search problems solvable in polynomial-time by an oracle Turing machine that has access to a type-2 **NP** oracle as well as its type-2 inputs.

Theorem 8 *[Kra95] Let Φ be a $\exists\forall$-sentence over a relational language L without $\leq$. If Φ fails in an infinite structure, then the type-2 problem Q_Φ is not in $(\mathbf{FP^{NP}})^2$.*

Since **ITER** $\in (\mathbf{FP^{NP}})^2$ trivially, the consequence of Theorem 8 is stronger than that of Main Theorem. However, it does not apply to search problems defined by $\exists$-sentences (which are also $\exists\forall$-sentences) over functional languages, which is the scope of Main Theorem. For example, Theorem 8 does not say anything about the complexity of **PIGEON**, since Φ_{PIGEON} is not over a relational language. In fact, **PIGEON** is in $(\mathbf{FP^{NP}})^2$: binary search asking 'does there exist $v > k$ witnessing Φ_{PIGEON}?' for various k yields a solution in polynomial-time.

5.1 Relative Complexity of PLS

Beame et al. [BCE+98] formulate a number of type-2 search problems. **LONELY** is based on the fact that there is no perfect matching in a graph with an odd number of vertices. More formally, **LONELY** is the type-2 search problem defined by the following $\exists$-sentence:

$$\alpha(0) = 0 \wedge (\forall x)[x = \alpha(\alpha(x))]$$
$$\supset (\exists x)[x \neq 0 \wedge x = \alpha(x)].$$

Note that the above sentence holds in every canonical structure V_n.

PIGEON, which we discussed in Section 5, is also from [BCE+98]. **SINK** and **SOURCE.OR.SINK** are also defined in the same paper, but we do not give their definitions here.

The above type-2 problems are formulated to capture the search classes defined by Papadimitriou in [Pap94b]. These classes are: **PPP**, **PPA**, **PPAD**, and **PPADS** (**PPADS** is given this name in [BCE+98]). They are known to contain natural problems, some of which are complete [Pap94b]. Their type-2 definitions are the following.

$$\begin{aligned} \mathbf{PPA} &= C(\mathbf{LONELY}) \\ \mathbf{PPAD} &= C(\mathbf{SOURCE.OR.SINK}) \\ \mathbf{PPADS} &= C(\mathbf{SINK}) \\ \mathbf{PPP} &= C(\mathbf{PIGEON}) \end{aligned}$$

Beame et al. show all possible separations and containments of these classes in a relativized world by obtaining all possible reducibilities and nonreducibilities among the type-2 problems.

However, **PLS** is not discussed in [BCE+98], and it was unknown how **PLS** relates to the other classes. Since the type-2 search problems of Beame et al. satisfy the premise of Main Theorem, the following is immediate from Main Theorem, Theorems 4 and Theorem 5, and the fact that **SOURCE.OR.SINK** many-one reduces to the other problems of [BCE+98].

Corollary 9 *There exists a relativized world in which* **PLS** *contains none of the following:* **PPA**, **PPAD**, **PPADS**, *and* **PPP**.

Corollary 9 may be interpreted as evidence that efficient local search heuristics are unlikely to exist for the problems in Papadimitriou's classes, which include problems related to cryptographic hash functions (**PPP**), finding

Nash equilibria (**PPAD**), and Sperner's lemma, Brouwer's fixed point theorem, and the Borsuk-Ulam theorem (**PPA**) [Pap94b].

5.2 Sentences with More Quantifiers

We generalize Main Theorem so that it applies to sentences with higher quantifier complexity. For $k \geq 1$, we say that a sentence is Σ_k if its prenex form has $k-1$ quantifier alternations with the outmost quantifier $\exists$.

Let Φ be a Σ_k-sentence, and let Φ_H be a $\exists$-sentence obtained by herbrandizing Φ. It is a well known fact that Φ logically implies Φ_H [Bus98b]. It is not hard to see that $Q_{\Phi_H} \leq_m Q_\Phi$. From this fact, the following is immediate:

Theorem 10 *Let $k \geq 1$ and Φ be a Σ_k-sentence over a first-order language L without the standard ordering $\leq$. If Φ fails in an infinite structure, then*

$$Q_\Phi \not\leq_T \textbf{ITER}.$$

It is interesting that Theorem 8 does not seem to generalize in the same way.

5.3 Proof of Main Theorem

Throughout this section, we fix the language L to be $L = \{0, \alpha\}$ and assume that Φ is an $\exists$-sentence over L of the form $(\exists x)\phi(x)$. The case with arbitrary language and arbitrary $\exists$-sentence is analogous to the current case.

For any $n \geq 1$, a partial function $\rho_n : V_n \mapsto V_n$ is called a *restriction*. Let $\rho = \{\rho_n\}_n$ be a family of restrictions. We denote by $(Q_\Phi)^\rho$ the type-2 search problem Q_Φ such that the oracle for α answers queries consistently with ρ, i.e., on instance $(1^n, \alpha)$, if $v \in dom(\rho_n)$, then the query '$\alpha(v)$' is answered with $\rho_n(v)$.

The size of restriction ρ_n is $|dom(\rho_n)| + |ran(\rho_n)|$ and is written $|\rho_n|$. We say that $\{\rho_n\}_n$ is a *polysize family* if $|\rho_n| \in n^{O(1)}$.

We say that a restriction $\rho_n : V_n \mapsto V_n$ contains a solution for Q_Φ if the defined part of ρ_n contains a witness to Φ in V_n.

A restriction $\rho_n : V_n \mapsto V_n$ is said to be *safe for* Φ if the following conditions are met: (1) there exists an infinite structure $\mathcal{K} = (K, \alpha_K)$ in which Φ fails; and (2) there exists a one-one mapping $f : V_n \mapsto K$ such that $\rho_n(v) = u$ implies $\alpha_K(f(v)) = f(u)$. Note that, if ρ_n is safe for Φ, then ρ_n does not force a solution for Q_Φ. We say that a family $\{\rho_n\}_n$ of restrictions is safe for Φ if ρ_n is safe for Φ for every n.

The separation proofs in [BCE$^+$98] involve the following model of computation. Define a *Search-tree* (or *S-tree*) T_n to be a directed tree whose internal nodes are labeled with *queries* '$\alpha(v)$' for some $v \in V_n$ and whose edges are labeled with *answers* 'u' for some $u \in V_n$. Each internal node specifies an oracle query to α, and if the answer is '$\alpha(v) = u$', then the outgoing edge with label 'u' should be taken, which leads to the node specifying the next query. This procedure terminates when a leaf node is reached. The leaf nodes are unlabeled. Krajicek [Kra95] has a similar construct called the *witness test tree*.

Note that, for every path P of T_n, there exists a corresponding restriction π_P specified by the queries and answers on P. We say that T_n solves Q_Φ on n if, for every path P of T_n, the corresponding restriction π_P contains a solution for Q_Φ.

Let $Depth_T(n)$ be the depth of T_n, i.e., the maximum length of paths from the root to a leaf node. A family $\{T_n\}_n$ of S-trees is said to be *poly-depth* if $Depth_T(n) \in n^{O(1)}$. Poly-depth families of S-trees constitute a nonuniform version of type-2 **FP**.

Special cases of the following are implicit in [Bus86, Kra95, BCE$^+$98, CK98, Mor01].

Lemma 11 *Let L be a language not containing $\leq$ and let Φ be an $\exists$-sentence over L that fails in an infinite structure. If $\{T_n\}_n$ is a poly-depth family of S-trees over L and $\rho = \{\rho_n\}_n$ is a safe, polysize family of restrictions, then, for all sufficiently large n, T_n contains a path P such that $\rho_n \cup \pi_P$ is safe for Φ.*

Note that the conclusion of Lemma 11 implies that T_n does not solve $(Q_\Phi)^\rho$ on n. A proof of

Lemma 11 is provided in Appendix C.

By Theorem 7, it suffices to prove $Q_\Phi \not\leq_m$ **ITER**. Assume for the sake of contradiction that $Q_\Phi \leq_m$ **ITER**. Let M be an oracle Turing machine that solves Q_Φ in polynomial-time by making one query to **ITER** and arbitrary many queries to α. Let $k(n) \in n^{O(1)}$ be the running time of M.

Claim 12 *There exists a polysize family $\{\rho_n\}_n$ of restrictions such that, for sufficiently large n, the following hold: (1) ρ_n is safe for Φ; and (2) ρ_n contains the answers to all the queries to α and* **ITER** *made by M on $(1^n, \alpha)$.*

Suppose Claim 12 holds and consider M on $(1^n, \alpha)$ for n sufficiently large. We answer all the queries to α and **ITER** according to ρ_n asserted to exist by the Claim. At the end of its computation, M is forced to output some $v \in V_n$ as a solution, although no solution is forced by ρ_n. Hence, after M outputs some v, we extend ρ_n to some α such that $\phi(v)$ does not hold in V_n. This completes the proof of Main Theorem.

It remains to prove Claim 12. Fix n sufficiently large so that all the necessary invocations of Lemma 11 hold, and let $k = k(n)$. We divide the computation of M into 3 phases: phase 2 is the **ITER**-query of M, and phase 1 and 3 consist of all the α-queries that are asked before and after the **ITER**-query, respectively.

For each phase, we construct a restriction μ_i such that (1) μ_3 extends μ_2, which extends μ_1; (2) μ_i is safe for Φ, and it contains the answers to all the queries that are asked in the first i phases of M; and (3) $|\mu_i| \in n^{O(1)}$.

In order to construct μ_1, let M' be an oracle Turing machine that simulates M until M writes the **ITER**-query, at which point M' halts. Thus, M' simulates phase 1 of M and asks at most k α-queries. Construct an S-tree T' from M' by extracting all possible sequences of α queries that M' asks. By Lemma 11 (with ρ_n empty), T' contains a path P such that π_P is safe for Φ. Let $\mu_1 = \pi_P$.

For phase 2, let $(1^m, \beta)$ be the **ITER**-query that M asks, when all preceding α-queries are answered according to μ_1. Our task is to construct μ_2 by extending μ_1 enough so that a solution for **ITER**$(1^m, \beta)$ is specified, while keeping μ_2 safe for Φ.

By definition of many-one reduction, $\beta : V_m \mapsto V_m$ is computable by an oracle Turing machine M_β in time polynomial in n using α as an oracle. For each $x \in V_m$, let $B(x)$ be the S-tree corresponding to the computations of M_β on x. We say a path P of S-tree $B(x)$ is *good* if P is consistent with μ_1 and $\mu_1 \cup \pi_P$ is safe. For each $x \in V_m$, let $Good_B(x)$ be the set of all good paths of $B(x)$. By Lemma 11, for all x, $Good_B(x)$ is not empty.

There are three cases to consider.

First Case: $Good_B(0^n)$ contains a path P such that the corresponding computation of M_β makes $\beta(0^n) = 0^n$. We set $\mu_2 = \mu_1 \cup \pi_P$ and return an arbitrary $v \in V_m$ to M as a solution for the **ITER**-query $(1^m, \beta)$.

Second Case: For some $x \in V_m$, $Good_B(x)$ contains a path P such that the corresponding computation of M_β makes $\beta(x) = y$ for some $y < x$. We set $\mu_2 = \mu_1 \cup \pi_P$ and return x as a solution for the **ITER**-query.

Third Case: the above two cases do not hold. Since the first case does not hold, every path in $Good_B(0^n)$ corresponds to a computation of M_β with $\beta(0^n) > 0^n$. Similarly, since the second case does not hold, every path in $Good_B(1^n)$ leads to $\beta(1^n) = 1^n$. Hence, by the least number principle, there exists $x \in V_m$ such that (1) $Good_B(x)$ contains a path P' that leads to $\beta(x) = y$ for some $y > x$; and (2) for all $z > x$, every path in $Good_B(z)$ leads to $\beta(z) = z$.

Let x, y, and P' be as in the preceding paragraph. Let $Better_B(y)$ as the set of paths P'' of $B(y)$ such that $\pi_{P''}$ is consistent with $\mu_1 \cup \pi_{P'}$ and $\mu_1 \cup \pi_{P'} \cup \pi_{P''}$ is safe for Φ. By Lemma 11, $Better_B(y)$ is not empty. Let P^* be any path in $Better_B(y)$. Set μ_2 to be $\mu_1 \cup \pi_{P'} \cup \pi_{P^*}$ and return x to M as a solution for its **ITER**-query. Note that x is a solution because $\beta(x) = y$ and $\beta(y) = y$. This concludes the construction of μ_2.

μ_3 is constructed in essentially the same way as μ_1. Let M'' be the machine that simulates phase 3 of M, and let T'' be the corresponding

S-tree. We find a path P of T'' that is consistent with restriction μ_2 and $\mu_2 \cup \pi_P$ is safe for Φ. By Lemma 11, such a path exists, and we set μ_3 to be $\mu_2 \cup \pi_P$.

Finally, let $\rho_n = \mu_3$. The resulting family $\{\rho_n\}_n$ satisfies the conditions in Claim 12.

6 Bounded Arithmetic

In his 1986 thesis Buss introduces theories S_2^i and T_2^i of Bounded Arithmetic, which have been shown to be closely linked to complexity classes in the Polynomial-time Hierarchy [Bus86, Bus98a, Kra95]. See Appendix D for basic definitions.

A connection between Bounded Arithmetic theories and the Polynomial-time Hierarchy is via the notions of definability of functions, relations, and search problems. Informally, a search problem is definable in theory T if its totality is provable in T.

Definition 13 *Let Q be a total search problem and T be a theory of Bounded Arithmetic. Q is Σ_i^b-*definable *in T if and only if for some Σ_i^b-formula $\phi(x, y)$,*

1. *For all $m, n \in \mathbb{N}$, $\phi(s_m, s_n)$ implies $n \in Q(m)$, where s_m and s_n are numerals representing x and y, respectively; and*
2. *$T \vdash (\forall x)(\exists y)\phi(x, y)$.*

Buss and Krajicek in [BK94] obtain a remarkable result which gives a logical characterization of **PLS** in Bounded Arithmetic. We state their result in terms of many-one reducibility.

Theorem 14 *A search problem Q is in* **PLS** *if and only if Q is Σ_1^b-definable in T_2^1.*

Let Φ be an $\exists$-sentence over an arbitrary language L. We denote by $\Phi^{\leq a}$ the $\Sigma_1^b(L)$-formula obtained by bounding all quantifiers of Φ by $2^{|a|}$, where a is the only free variable of the formula. We say that a language L is BA-disjoint if $L - \{0\}$ is disjoint with the language L_{BA} of Bounded Arithmetic. The following is a consequence of Theorems 4 and a relativized version of Theorem 14.

Theorem 15 *Let Φ be an $\exists$-sentence over a BA-disjoint language L, and Q_Φ be the corresponding type-2 search problem. If $T_2^1(L)$ proves $\Phi^{\leq a}$, then $Q_\Phi \leq_m$* **ITER**.

Thus, by showing $Q_\Phi \not\leq_m$ **ITER**, one obtains the independence of $\Phi^{\leq a}$ from $T_2^1(L)$.

From Main Theorem and Theorem 15, we obtain the following.

Theorem 16 *Let Φ be an $\exists$-sentence over a BA-disjoint language L. If Φ fails in an infinite structure, then*

$$T_2^1(L) \nvdash \Phi^{\leq a}.$$

Theorem 16 is a special case of Riis's first finitisation principle in [Rii93]. It states that, if a sentence Φ over a BA-disjoint language L fails in an infinite structure, then $\Phi^{\leq a}$ is not provable in $S_2^2(L)$. Riis proves it by a model-theoretic argument, and Krajicek later gives an alternative proof which uses Theorem 8.

Although Theorem 16 does not by itself give any new independence result for $T_2^1(L)$, some of the known independence results for $T_2^1(L)$ do follow from it.

For example, consider any $\exists$-sentence Φ defining any of **PIGEON**, **LONELY**, **SINK**, and **SOURCE.OR.SINK**. From Theorem 16 it follows that the $\Sigma_1^b(L)$-formula $\Phi^{\leq a}$ is not provable in $T_2^1(L)$. Note that these formulas have been known to be independent of the whole $S_2(L)$, which is much larger than $T_2^1(L)$ [CK98].

Two more combinatorial principles are shown to be independent of $T_2^1(L)$ in [CK98]. One is a functional version of the *weak pigeonhole principle*, and the other is called the *generalized iteration principle*. The independence of these statements also follow from Theorem 16.

7 Concluding Remarks

We have shown that none of the type-2 problems of [BCE+98] is reducible to **ITER**. A natural question is whether **ITER** is reducible to any of these. In a joint work with Buresh-Oppenheim, we have been able to show that

ITER is not reducible to **LONELY** [BOM03], and hence $(\mathbf{PLS})^A \not\subseteq (\mathbf{PPA})^A$ for some oracle A. However, it is still open whether **ITER** is reducible to **PIGEON**.

It is also open whether analogues of Theorem 7 hold with respect to the type-2 problems of [BCE+98]. This is related to the question whether Papadimitiou's classes are closed under Turing reducibility.

By a result of Buss and Krajicek [BK94], $\mathbf{FP}^A \not\subseteq \mathbf{PLS}^A$ for some oracle A. However, in the unrelativized context, $\mathbf{PLS} \subseteq \mathbf{FP}$ is consistent with the current knowledge of complexity theory. It would be interesting to derive a complexity-theoretic consequence of $\mathbf{PLS} \subseteq \mathbf{FP}$.

Acknowledgements. We wish to thank S. Riis for helpful suggestions on [Mor01] that motivated the Main Theorem.

References

[BCE+98] P. Beame, S. A. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57:3–19, 1998.

[BK94] S. R. Buss and J. Krajicek. An application of Boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–27, 1994.

[BOM03] J. Buresh-Oppenheim and T. Morioka. In preparation, 2003.

[Bus86] S. R. Buss. *Bounded Arithmetic*. Bibliopolis, 1986.

[Bus98a] S. R. Buss. First-order proof theory of arithmetic. In S. R. Buss, editor, *Handbook of proof theory*, pages 79–147. Elsevier Science, 1998.

[Bus98b] S. R. Buss. An introduction to proof theory. In S. R. Buss, editor, *Handbook of proof theory*, pages 1–78. Elsevier Science, 1998.

[CIY97] S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity. *Information and Computation*, 137(2):159–170, 1997.

[CK98] M. Chiari and J. Krajicek. Witnessing functions in bounded arithmetic and search problems. *The Journal of Symbolic Logic*, 63:1095–1115, 1998.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[Kra95] J. Krajicek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1995.

[Mor01] T. Morioka. Classification of Search Problems and Their Definability in Bounded Arithmetic. Master's thesis, University of Toronto, 2001. Also available as ECCC technical report TR01-82 at http://www.eccc.uni-trier.de.

[MP91] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81:317–324, 1991.

[Pap94a] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pap94b] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48:498–532, 1994.

[Rii93] S. Riis. Making infinite structures finite in models of second order bounded arithmetic. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational*

Complexity, pages 289–319. Oxford University Press, 1993.

[Rii01] S. Riis. A complexity gap for tree resolution. *Computational Complexity*, 10:179–209, 2001.

[Tow90] M. Townsend. Complexity for type-2 relations. *Notre Dame Journal of Formal Logic*, 31(2):241–262, 1990.

[Yan97] M. Yannakakis. Computational complexity. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. John Wiley and Sons Ltd., 1997.

A Proof of Theorem 6

Theorem 6 **ITER** *and* **BTROOT** *are many-one reducible to each other.*

BTROOT $\leq_m$ **ITER** is trivial. For the other direction, let $(1^n, \alpha)$ be an instance of **ITER** and construct an instance $(1^{2n}, \gamma, \beta_1, \beta_2)$ as follows.

First, consider every $z \in V_{2n}$ as consisting of two components z^1 and z^2 such that $z = z^1 \circ z^2$. $\gamma : V_{2n} \mapsto V_{2n}$ is defined as

$$\gamma(z^1 \circ z^2) = \begin{cases} z^1 \circ z^2 & \text{if } \alpha(z^1) = z^1 \\ z^1 \circ (z^2 + 1) & \text{if } \alpha(z^1) \neq z^1 \\ & \text{and } z^2 \neq 1^n \\ (\alpha(z^1)) \circ z^1 & \text{if } \alpha(z^1) \neq z^1 \\ & \text{and } z^2 = 1^n \end{cases}$$

where $z^2 + 1$ is the lexicographically next string of z^2.

Let us describe the idea behind the definition of γ. For each $v \in V_n$, we associate a lexicographically ordered sequence $v \circ 0^n, \ldots, v \circ 1^n$. If $\alpha(v) = v$, then γ is the identity mapping on the sequence. Otherwise, the sequence forms a chain in which $v \circ i$ is mapped to $v \circ (i+1)$, for all $i \in V_n - \{1^n\}$. Only the 'top element' $v \circ 1^n$ of the chain is mapped to something outside of the chain, namely, the vth element of the chain corresponding to $\alpha(v)$.

Note that, for most of $z \in V_{2n}$, there are at most two nodes that can be mapped by γ to z: one is $z^1 \circ (z^2 - 1)$ and the other is $z^2 \circ 1^n$. Thus, β_1 and β_2 are defined as follows:

$$\beta_1(z) = \begin{cases} z^1 \circ z^2 & \text{if } z^2 = 0^n \vee \alpha(z^1) = z^1 \\ z^1 \circ (z^2 - 1) & \text{otherwise} \end{cases}$$

$$\beta_2(z) = \begin{cases} z^2 \circ 1^n & \text{if } \gamma(z^2 \circ 1^n) = z \\ z^1 \circ z^2 & \text{otherwise} \end{cases}$$

Note that all γ, β_1, and β_2 are polynomial-time computable with oracle α. The correctness of the reduction follows from the following five properties:

1. if $\gamma(0^{2n}) = 0^{2n}$, then $\alpha(0^n) = 0^n$;
2. if $\gamma(z) < z$, then $\alpha(z^1) < z^1$;
3. β_1 is not increasing anywhere, and if $\beta_2(z) > z$ then $z^2 > \alpha(z^2)$;
4. for all $x \neq y$, $\gamma(x) = y$ iff either $\beta_1(y) = x$ or $\beta_2(y) = x$;
5. if $z < \gamma(z)$ and $\gamma(z) = \gamma(\gamma(z))$, then $z^1 < \alpha(z^1)$ and $\alpha(z^1) = \alpha(\alpha(z^1))$.

It is not hard to verify the above properties.

B Proof of Theorem 7

Theorem 7 *Let Q be a search problem (type-1 or type-2). $Q \leq_m$* **ITER** *if and only if $Q \leq_T$* **ITER**.

Let Q be a type-1 search problem and assume that $Q \leq_T$ **ITER**. We show that $Q \leq_m$ **ITER**. The case with type-2 Q is entirely analogous.

Let M be an oracle Turing machine that solves Q by making queries to **ITER** and $k(n) \in n^{O(1)}$ be the number of **ITER**-queries that M makes on inputs of length n. Assume without loss of generality that the type-1 argument of every **ITER**-query is $1^{m(n)}$.

Let n be arbitrary and $m = m(n)$ and $k = k(n)$. Consider every element $v \in V_{km}$ as

the concatenation of k components of length m, i.e., $v = v^1 \circ v^2 \circ \ldots \circ v^k$, where each $v^i \in V_m$.

The first **ITER**-query of M depends only on input x, but all subsequent queries depend on the answers to the preceding queries. For $1 \leq i \leq k$, we write $\gamma[x, s_1, \ldots, s_i]$ to denote the type-2 argument of the ith **ITER**-query that M asks when, for each $j < i$, the jth query is answered by s_j. We do not require that s_j be actually a solution for the jth query. Note that x, $s_1, \ldots, s_i$ are not arguments for $\gamma[x, s_1, \ldots, s_i]$, which is a mapping from V_m to V_m.

Define a relation Sol as follows. For $i \in \{1, \ldots, k\}$ and $v \in V_{km}$, $Sol(i, v)$ holds iff, for all $j \leq i$, v^j is a solution for $\mathbf{ITER}(1^m, \gamma[x, v_1, \ldots, v_{j-1}])$. Let $MaxSol(v)$ be the maximum i such that $Sol(i, v)$ holds. Clearly, $MaxSol$ is polynomial-time computable in n with access to α.

For $v \in V_{km}$, define $\beta(v)$ to be

$$(v^1 \circ \ldots \circ v^l) \circ \gamma[x, v^1, \ldots, v^l] \circ (v^{l+2} \circ \ldots \circ v^k),$$

where $l = MaxSol(v)$. β leaves every component of v unchanged except the $i+1$st component, to which $\gamma[x, v^1, \ldots, v^i]$ is applied. Note that β is computable by a Turing machine in time polynomial in n with access to α.

It is not hard to see that, if v is solution for $\mathbf{ITER}(1^{km}, \beta)$, then $Sol(i, v)$ holds for every $1 \leq i \leq k$.

Finally, the following machine M' is a many-one reduction from Q to **ITER**: given x, M' asks an **ITER** query $(1^{km}, \beta)$, where β is presented as, say, a polynomial-time Turing machine. Upon receiving a solution v, M' simulates M on x, using the ith component v^i as a solution for the ith query of M. When M halts, M' halts with the same output, which is a solution for $Q(x)$.

C Proof of Lemma 11

Lemma 11 *Let L be a language not containing $\leq$ and let Φ be an $\exists$-sentence over L that fails in an infinite structure. If $\{T_n\}_n$ is a poly-depth family of S-trees over L and $\rho = \{\rho_n\}_n$ is a safe, polysize family of restrictions then, for all sufficiently large n, T_n contains a path P such that $\rho_n \cup \pi_P$ is safe for Φ.*

For simplicity, let $L = \{0, \alpha\}$ and Φ be of the form $(\exists x)\phi(x)$. More general cases are proven analogously. Fix n sufficiently large so that $|V_n| = 2^n >> Depth_T(n) + |\rho_n|$. Since ρ_n is safe for Φ, there exists an infinite structure $\mathcal{K} = (K, \alpha_K)$ in which Φ fails and a one-one mapping $f : V_n \mapsto K$ such that $\rho_n(v) = u$ implies $\alpha_K(f(v)) = f(u)$.

When $\mu : V_n \mapsto V_n$ is a restriction, we say that μ *touches* $v \in V_n$ if $v \in dom(\mu) \cup im(\mu)$. Let $g : V_n \mapsto K$ be a partial mapping defined as

$$g(v) = \begin{cases} f(v) & \text{if } \rho_n \text{ touches } v; \text{ and} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We incrementally extend g while traversing T_n, applying the following procedure at each node, starting with the root of T_n. Let '$\alpha(v)$' be the label of the current node, and let P denote the path from the root of T_n to the parent node of the current node. Let μ denote the restriction $\rho_n \cup \pi_P$. We maintain the following invariants: (i) g is one-one; (ii) for all $v \in V_n$, $g(v)$ is defined iff μ touches v; and (iii) for all $u, v \in V_n$, if $\mu(v) = u$ then $g(v)$ and $g(u)$ are defined and $\alpha_K(g(v)) = g(u)$.

Two cases arise. Case (1): $v \in dom(g)$. Then there exists $u_K \in K$ such that $\alpha_K(g(v)) = u_K$. If $u_K \in im(g)$, then take the outgoing edge labeled with $g^{-1}(u_K)$. If $u_K \notin im(g)$, then choose an arbitrary $u \in V_n$ that neither ρ_n nor π_P touches, and set $g(u) := u_K$. Take the outgoing edge labeled with u. Case (2): $v \notin dom(g)$. Then pick an arbitrary $v_K \in K \backslash im(g)$, and let $u_K = \alpha_K(v_K)$. We set $g(v) := v_K$. If $u_K \notin im(g)$, then choose an arbitrary $u \in V_n$ that neither ρ_n nor π_P touches, and set $g(u) := u_K$. Take the outgoing edge labeled with $g^{-1}(u_K)$.

It is not hard to verify that the invariants hold throughout the procedure.

When we reach a leaf of T_n by applying the above procedure, let P denote the path that has been traversed. The invariants imply that P is a path of T_n such that $\rho_n \cup \pi_P$ is safe for Φ.

D Bounded Arithmetic

In his 1986 thesis Buss introduces theories S_2^i and T_2^i of Bounded Arithmetic, which have been shown to be closely linked to complexity classes in the Polynomial-time Hierarchy [Bus86, Bus98a, Kra95]. These theories are defined over the language of Bounded Arithmetic

$$L_{BA} = \{0, S, +, \cdot, \lfloor \frac{x}{2} \rfloor, |x|, \#, \leq\},$$

where 0 is a constant, S is the successor function, $|x| = \lceil \log_2 x \rceil$ denotes the binary length of x, and $x\#y = 2^{|x|\cdot|y|}$ is the smash function. **BASIC** denotes the set of axioms that define the meaning of the nonlogical symbols in L_{BA}. Finally, we write $a \leq b$ as a shorthand for $\neg(a > b)$.

A quantifier is said to be *sharply bounded* if it is of the form $(\exists x \leq |t(\overline{a})|)$ or $(\forall x \leq |t(\overline{a})|)$, where t is a term in the language L_{BA}. A quantifier is said to be *bounded* if it is of the form $(\exists x \leq t(\overline{a}))$ or $(\forall x \leq t(\overline{a}))$.

Σ_1^b is the class of formulas whose bounded quantifiers are all existential (arbitrary sharply-bounded quantifiers are allowed). More generally, for $i \geq 1$, Σ_i^b is the class of formulas with $i-1$ alternations of bounded quantifiers, starting with the bounded existential quantifiers. Every Σ_i^b-formula represents an $\mathbf{\Sigma}_i^p$ relation, and conversely.

Let Φ be a set of formulas. The Φ–IND axioms are the formulas

$$A(0) \wedge (\forall x)[A(x) \supset A(Sx)] \supset (\forall x)A(x)$$

for all formulas $A \in \Phi$. Similarly, Φ–$LIND$ axioms are the formulas

$$A(0) \wedge (\forall x)[A(x) \supset A(Sx)] \supset (\forall x)A(|x|)$$

for all $A \in \Phi$.

S_2^i is the theory axiomatized by **BASIC** plus Σ_i^b–$PIND$, and T_2^i is the theory axiomatized by **BASIC** plus Σ_i^b–IND. These are called BA theories.

BA theories are relativized in the following way. Let $\alpha \notin L_{BA}$ be a function symbol. $\Sigma_i^b(\alpha)$ is defined over language $L_{BA} \cup \{\alpha\}$ similarly to Σ_i^b with a restriction that α does not appear in the bound of a quantifier. Then relativized theory $S_2^i(\alpha)$ and $T_2^i(\alpha)$ are axiomatized by **BASIC** plus appropriate induction axioms for $\Sigma_i^b(\alpha)$-formulas. Note that the relativized BA theories have no axiom defining the meaning of α.

For an arbitrary language L, $S_2^i(L)$ and $T_2^i(L)$ are obtained similarly over language $L_{BA} \cup L$.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'