SOME RELATIONS BETWEEN SUBSYSTEMS OF ARITHMETIC AND COMPLEXITY OF COMPUTATIONS

PAVEL PUDLÁK

ABSTRACT. We shall introduce a special mode of interactive computations of optimal solutions to optimization problems. A restricted version of such computations was used in [KPT] to show that $T_2^i = S_2^{i+1}$ implies $\Sigma_{i+2}^p = \Pi_{i+2}^p$. Here we shall reduce the question whether $T_2^i = S_2^i$ to a question about interactive computations (in a more general sense) of some optimization problems.

1. INTRODUCTION

We shall consider fragments of bounded arithmetic S_2. This is a first order theory of arithmetic, where the induction schema is restricted to bounded formulae. Our main reason for studying such systems is their close relation to low level computational complexity. We hope that eventually this research will bring new insight into the problems in complexity theory. Ideally, we would like to show some independence results for such theories and sentences stating some unknown relations between complexity classes. In order to be able to prove such results, we have to understand better the mutual relation between the complexity theory and such logical theories. This paper attempts to make another step in this direction.

The first system for bounded arithmetic was proposed and studied by Parikh [Pa]. Nowadays his system is known as $I\Delta_0$; it is Peano Arithmetic with induction restricted to bounded arithmetical formulae. This and related systems have been extensively studied by Paris and Wilkie (see [PW] for a survey paper). The system S_2 and an equivalent system T_2 were introduced by Buss [B1]. These systems are conservative extensions of $I\Delta_0 + \Omega_1$, where Ω_1 is $\forall x \exists y (y = 2^{\lceil \log_2(x+1) \rceil})$. The richer language of S_2 and T_2 enables one to define natural fragments S_2^i and T_2^i, $i = 1, 2, \ldots$. The definition of S_2^i and T_2^i is motivated by Stockmeyer's Polynomial Hierarchy. This hierarchy is a natural extension of the classes $\mathcal{P}$, $\mathcal{NP}$, $co\mathcal{NP}$ in

Y. N. Moschovakis (ed.), *Logic from Computer Science*

a much similar way as the arithmetical hierarchy is an extension of classes *recursive sets, recursively enumerable sets, complements of recursively enumerable sets*, etc. It is an open problem whether S_2 is finitely axiomatizable. This problem is equivalent to the statement that S_2 collapses to some fragment S_2^i. Similarly it is an open problem whether Polynomial Hierarchy collapses to some level.

In [KPT] we showed that if S_2 is finitely axiomatizable, then Polynomial Hierarchy collapses. This was done by proving

$$T_2^i = S_2^i \Rightarrow \Sigma_{i+2}^p = \Pi_{i+2}^p$$

(where Σ_{i+1}^p and Π_{i+2}^p are levels in the Polynomial Hierarchy), and using the fact that we have the following inclusions

$$T_2^0 \subseteq S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \dots .$$

Hence assuming a plausible conjecture that Polynomial Hierarchy does not collapse, the odd inclusions are strict. Here we try to do a similar thing with the even inclusions. We shall present a conjecture on certain computations which implies that also the odd levels are strict. This is a conjecture about a new concept and, unfortunately, we are not able to reduce it to any problem in complexity theory which has been considered before.

It should be stressed that the motivation for showing $S_2^i \neq T_2^i$ is not only to clear up the remaining problems. A strong conservation result has been proved for pairs T_2^i and S_2^{i+1} in [B2]. If there were some partial conservativity also between pairs S_2^i and T_2^i, we would have some partial conservativity of the whole system S_2 over its fragments S_2^i. This seems rather unlikely. The separation of S_2^i from T_2^i could be the first step toward showing that there is no such conservativity. For related results see also Krajíček's paper [Kr] in these proceedings.

2. Fragments of Bounded Arithmetic

We shall use fragments of systems of bounded arithmetic S_2 and T_2 of Buss [B1]. Each system has a finite set of basic open axioms and an axiom schema of induction. We consider two schemata. Ordinary induction IND:

$$\varphi(0) \;\&\; \forall x(\varphi(x) \to \varphi(x+1)) \to \forall x \varphi(x);$$

and *polynomial induction* PIND:

$$\varphi(0) \;\&\; \forall x(\varphi(\lfloor x/2 \rfloor) \to \varphi(x)) \to \forall x \varphi(x).$$

S_2 is the system with the basic axioms and the schema PIND for all bounded formulae; T_2 is the system with the basic axioms and the usual schema of induction IND for all bounded formulae.

The particular choice of primitive notions and basic axioms describing the primitive notions is not very important, hence we shall not give a precise definition, instead we shall describe the most important properties that we need:

(1) the language contains a finite number of polynomial time computable functions and predicates;
(2) there are natural classes of formulae defining the sets in the Polynomial Hierarchy.

The reason for extending the usual language of arithmetic is that we want to have naturally defined fragments of these theories. If we did not extend the language, the bottom fragments would not have nice properties. Furthermore, the schemata of induction for bounded formulae are not strong enough to pove the existence of functions that grow faster than the *terms* of the language. Thus it is more convenient to add functions with higher growth rate than extend the axiomatization, which is Π_1^0, by a Π_2^0 axiom. The higher growth rate is needed in order to be able to formalize polynomial time computations. If a sequence s of length k is encoded by a binary expansion of a number n, then n is about 2^k. If we need another sequence of length k^2, then we need a number of size 2^{k^2}, which is about $2^{(\lceil \log_2(n+2)\rceil)^2}$. That is why we need such a function. Note that there is an alternative approach which is based on the sequence as the basic primitive concept instead of the concept of the number. This has been considered by Cook [C], (however his theories PV and $PV1$ are quantifier free).

The classes of bounded formulae Σ_i^b and Π_i^b are defined as follows. First choose a suitable class $\Sigma_0^b (= \Pi_0^b)$ whose formulae define sets in $\mathcal{P}$. Then we define Σ_i^b and respectively Π_i^b as the classes of formulae with the corresponding prefix of *bounded* quantifiers follows by a Σ_0^b formula. The formulae in Σ_0^b are defined using *sharply bounded quantifiers*. These are bounded quantifiers where the bound is always the logarithm of a term. Thus we shall use also the function $\lceil \log_2(x+1)\rceil$, which is denoted by $|x|$. For $i > 0$, the formulae in Σ_i^b (respectively in Π_i^b) define just the sets in Σ_i^p (respectively in Π_i^p). Using this relation we shall sometimes identify Σ_i^p sets with their Σ_i^b definitions. Σ_0^b formulae, as they are usually defined, do not define all sets in $\mathcal{P}$. This is rather inconvenient and complicates the statements of the theorems. Therefore in this paper we shall assume that Σ_0^b is a suitable class of formulae that define just the sets in $\mathcal{P}$. Such a class can be constructed by formalizing polynomial time Turing machine computations. This change influences only the weakest fragment that we consider T_2^0: now in T_2^0 we can define all polynomial time computable functions. To get the full symmetry, we shall use another convention which is not quite standard: we shall sometimes denote $\mathcal{P}$ by Σ_0^p and Π_0^p.

Now we can define fragments of bounded arithmetic: S_2^i is S_2 with PIND restricted to Σ_i^b formulae, T_2^i is T_2 with induction restricted to Σ_i^b formulae, $i \geq 0$. Thus, for instance, we can think of T_2^0 as "induction for $\mathcal{P}$", and of S_2^1 as "polynomial induction for $\mathcal{NP}$".

Often it will be more convenient to use the following schema LIND (a form of the least number principle) instead of PIND:

$$\neg\varphi(x,0) \vee \exists t < |x|(\varphi(x,t) \ \&\ \neg\varphi(x,t+1)) \vee \varphi(x,|x|).$$

Fragments S_2^i can be axiomatized by this schema for Σ_i^b formulae. It has been proven in [B1] that

$$T_2^0 \subseteq S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \dots ,$$

hence

$$S_2 = \cup S_2^i \equiv \cup T_2^i = T_2.$$

Many facts about the subsystems $I\Sigma_i^0$ of Peano Arithmetic transfer to fragments S_2^i and T_2^i (e.g. $\mathrm{PIND}\,\Sigma_i^b \equiv \mathrm{PIND}\,\Pi_i^b$), but not all. In particular the proof that Peano Arithmetic is not finitely axiomatizable breaks down in the new context. To prove that Peano Arithmetic is not finitely axiomatizable one shows that

$$I\Sigma_{i+1}^0 \vdash \mathrm{Con}(I\Sigma_i^0)$$

(where Con denotes the consistency), and uses Gödel's Theorem to show that

$$I\Sigma_i^0 \vdash \mathrm{Con}(I\Sigma_i^0)$$

does *not* hold. While Gödel's Theorem holds for fragments of bounded arithmetic (see [B1]), S_2 does not prove even the consistency of basic axioms; (there are stronger results in this direction in [B1], [PW], [Pu], [T]).

3. Optimization Problems

The aim of this paper is to show that there is a close connection between fragments S_2^i and T_2^i on the one hand and certain interactive computations of solutions to optimization problems on the other hand. Here we introduce some basic terminology on optimization problems.

Let $C(x, y)$ be a binary relation, let ρ be a function; both are defined on natural numbers. We shall think of x as an input. For an input x, any y such that $C(x, y)$ holds will be called *a feasible solution to* x, (put otherwise, C is the condition that determines what is feasible for x). Function ρ measures how good a solution is: if y and y' are feasible solutions, then y' is better than y if $\rho(y) < \rho(y')$. If there is no better solution to x than y, then y is called an *optimal solution.* For a pair C, ρ to determine an *optimization problem* we shall assume two purely technical conditions, which will greatly simplify the exposition:

(1) 0 is a feasible solution to any x;
(2) if y is a feasible solution to x, then $y \leq x$.

We shall, of course, assume that all finite objects are encoded as numbers.

From the practical point of view only problems where C is in $\mathcal{P}$ and ρ is polynomial time computable are interesting. Most of the $\mathcal{NP}$-problems come from optimization. Let us consider two examples.

(1) CLIQUE
$C(x, y) \equiv$ "y is a clique in graph x",
$\rho(y) =$ "the size of y",
(0 is assumed to be the code of the empty clique).
(2) TRAVELLING SALESPERSON
$C(x, y) \equiv$ "y is a tour in graph x whose edges are labelled by numbers (or $y = 0$)",
$\rho(y) =$ "sum of all the labels of the graph minus the length (i.e. the sum of the labels) of the tour" (and $\rho(0) = 0$).

It seems that there is some inherent difference between the two problems. In the first case the range of ρ on feasible solutions to x is bounded by a polynomial in the *size* of x, (in fact it is less than the size of x). In the second case the range may be exponentially large in the size of x, since numbers up to $2^n - 1$ have length less than or equal to n in binary notation. For usual deterministic polynomial time computations this makes little difference, since we know that both problems are $\mathcal{NP}$-hard, but if we have some extra information available the first type of a problem is more likely to be solvable.

It is possible to define a hierarchy of optimization problems according to the size of the range of ρ. However, for this paper we need only the distinction between the polynomial size range and the exponential size range. We shall call them *type* 1 and *type* 2 optimization problems respectively. Thus *type* 2 are in fact all optimization problems and we use this term only to stress the difference. For reasons of symmetry which will be apparent later, we define also *type* 0: the problems in which the range of ρ is uniformly bounded by a constant.

Optimization problems with C in $\mathcal{P}$ will be related to fragments T^0_2, S^2_1, T^1_2. For higher fragments we shall need C in Π^p_{i-1}, while ρ will always be polynomial time computable.

4. Computations with Counterexamples

Let an optimization problem $C(x, y), \rho$ be given. Suppose a person called *STUDENT* is to determine an optimal solution to x and suppose he can use the help of another person called *TEACHER*. *STUDENT* has limited ability, (in the simplest case he can perform deterministic polynomial time computations), while *TEACHER* knows everything about the given problem. *STUDENT* can ask questions of the form:

Is y an optimal solution to x?

TEACHER must answer correctly and, moreover, if her answer is negative, she must produce a counterexample which is some better feasible solution to x. The aim of *TEACHER* is to test *STUDENT*, so she can choose counterexamples which convey as little information as possible to *STUDENT*.

Formally we would model *STUDENT* as a mutlitape Turing machine where the queries of *STUDENT* and answers of *TEACHER* will appear on a special tape of the machine. The queries are computed by the machine, while the answers come from outside following an arbitrary strategy. We define that *STUDENT* (i.e. a given Turing machine) *solves* the optimization problem, if for every x and every strategy of *TEACHER*, *STUDENT* computes some optimal solution to x. Thus *TEACHER* does not act as an oracle in the ordinary sense, she is rather a person in a two player game. Further we shall modify this model by allowing *STUDENT* to use an ordinary racle from some class Σ^p_i and by restricting the number of queries posed to *TEACHER*. However we shall always assume that the number of steps in the computation is bounded by a polynomial.

There is a *trivial strategy* for *STUDENT* (which is often used by stupid students) according to which his first conjecture is 0 and then *STUDENT* just repeats the answers of *TEACHER*. Clearly, for optimization problems of type 1, this strategy produces always a solution. We conjecture that there is no strategy for *STUDENT* in general (i.e. for type 2). In fact, if we measure the complexity by the number of queries that *STUDENT* must ask, then it is plausible that there is no better strategy for hard problems than the trivial one.

We define two types of computations with counterexamples:

type 0: the number of queries is bounded by a constant;

type 1: the number of queries is unbounded, (implicitly it is always bounded by a polynomial, since all computations are bounded by a polynomial).

We have already noticed that type 1 computations solve type 1 problems; the same is true for types 0.

Let us consider for a moment computations with counterexamples without an additional Σ_i^p oracle, and suppose we want to find an optimal solution for a predicate $C(x,y)$ which is in $\mathcal{P}$. If we want to use the usual oracle computation instead of the counterexample computation, we can replace *TEACHER* by a Σ_1^p oracle as follows. The query

Is y optimal?

is Π_1^p, and if the answer is *not*, we can ask a Σ_1^p oracle about the bits of a better feasible solution. Hence if optimal solutions can be computed with counterexamples, then they can be computed by computations with a Σ_1^p oracle. However, the computations with Σ_1^p oracles are too strong: it is an easy exercise to show that any (i.e. type 2) problems can be solved using such computations. So it is important that *STUDENT* is allowed to ask only about feasible solutions to x when he is computing an optimal solution to x.

We shall also use computations with counterexamples in a more general situation. Let $B(x,y,z)$ be a ternary predicate, let x be given. Now the aim of *STUDENT* is to find some y such that $\forall z \leq xB(x,y,z)$. Again *STUDENT* can produce a conjecture y and ask *TEACHER* whether $\forall z \leq xB(x,y,z)$ is true. If it is not true, *TEACHER* must give to *STUDENT* some z such that $z \leq x$ & $\neg B(x,y,z)$. Such a z will be called a counterexample. If $C(x,y), \rho(y)$ is an optimization problem, then we define $B(x,y,z)$ by

$$B(x,y,z) \equiv C(x,y) \ \& \ (\rho(y) < \rho(z) \rightarrow \neg C(x,z));$$

thus $\forall z \leq x B(x,y,z)$ expresses that y is an optimal solution to x. Note that if C is in $\Sigma_i^b \cup \Pi_i^b$, then B is in $\Sigma_{i+1}^b \cap \Pi_{i+1}^b$. Now this more general definition of computatios allows *STUDENT* to ask also about elements which are not feasible solutions to C. But this is no real advantage for *STUDENT*, since *STUDENT* can always test himself whether y is a feasible solution, and if it is not, then clearly $z = 0$ is a counterexample and he can use it as a possible answer to *TEACHER*.

5. The Existence of Optimal Solutions in Fragments of Bounded Arithmetic

We consider optimization problems of the form $C(x,y), \rho(y)$ where $C(x,y)$ is Π_i^p and ρ is polynomial time computable. We shall suppose that $C(x,y)$ is defined by a Π_i^b formula, $\rho(y)$ is defined by a Σ_0^b formula, and formulae $C(x,0)$ and $C(x,y) \rightarrow y \leq x$ are provable in the fragment in question. If $C(x,y), \rho(y)$ is a type 1 optimization problem, we shall require that

$$C(x,y) \rightarrow \rho(y) < |x|$$

is also provable.

Proposition 1. *The following holds modulo the basic axioms for $i \geq 0$.*

(1) PIND Σ_{i+1}^b *is equivalent with the schema saying that every Π_i^p optimization problem of type 1 has an optimal solution;*
(2) IND Σ_{i+1}^b *is equivalent with the schema saying that every Π_i^p optimization problem of type 2 has an optimal solution.*

Proof. We shall prove only (1), since (2) is similar. We shall use the equivalent schema LIND Σ_{i+1}^b. Let $C(x,y), \rho(y)$ be given, let $C(x,y)$ be in Π_i^p. Assume that $C(x,y) \rightarrow \rho(y) < |x|$ is provable in the base theory. Take $\varphi(x,t)$ defined by

$$\varphi(x,t) \equiv \exists y \leq x (C(x,y) \ \& \ t \leq \rho(y)).$$

From LIND Σ_{i+1}^b we get

$$\neg\varphi(x,0) \vee \exists t < |x| (\varphi(x,t) \ \& \ \neg\varphi(x,t+1)) \vee \varphi(x,|x|),$$

which just expresses the existence of an optimal solution.

Proving the other direction assume that we are given some Σ^b_{i+1} formula $\Psi(x,u)$. We want to derive an instance of PIND for $\Psi(x,u)$, where u is the induction variable and x is a parameter. It is easily seen that we can consider only formulae of the form

$$\Psi(x,u) \equiv \exists v((u,v) \leq x \ \& \ \psi(x,u,v)),$$

where $(-,-)$ is the pairing function and ψ is Π^b_i, since PIND for such formulae proves the full schema PIND Σ^b_{i+1}. Let $C(x,y)$ and $\rho(y)$ be defined by

$$\begin{aligned} C(x,y) &\equiv (y \leq x \ \& \ \psi(x,(y)_0,(y)_1) \ \& \ (y)_0 \leq |x|) \vee y = 0, \\ \rho(0) &= 0, \\ \rho(y) &= (y)_0 \text{ for } y > 0, \end{aligned}$$

where $(y)_0$ and $(y)_1$ are the decoding functions for the pairing function. Suppose we have

$$\Psi(x,0) \text{ and } \forall u < |x|(\Psi(x,u) \rightarrow \Psi(x,u+1)).$$

If the problem C, ρ has an optimal solution y for a given x, then it must be such that $(y)_0 = |x|$, hence we have $\Psi(x,|x|)$. Thus we have shown PIND for Ψ. □

Note that to prove that a type 0 optimization problem has an optimal solution, we do not need *any* induction.

6. Separation of Fragments

It has been noted quite early in the history of proof theory, that if we prove a sentence $\forall x \exists y \varphi(x,y)$ in some theory, then we have some information about how difficult is to find some y for a given x such that $\varphi(x,y)$. In his fundamental paper [Pa] Parikh showed that if $\forall x \exists y \varphi(x,y)$, with φ bounded, is provable in $I\Delta_0$, then y can be bounded by a polynomial in x, consequently it can be computed in linear space. Buss [B1] proved a theorem about S_2 which gives essentially more information. He proved the following theorem.

Theorem 1. *Let $i \geq 0$, let $\varphi(x, y)$ be Σ^b_{i+1}, and suppose*

$$S^{i+1}_2 \vdash \forall x \exists y \varphi(x, y).$$

Then there exists a function f computable in polynomial time with a Σ^p_i oracle such that

$$\mathbb{N} \models \forall x \varphi(x, f(x)).$$

(Actually he proved more: under the same assumption $S^{i+1}_2 \vdash \forall x \varphi(x, f(x))$.)

We are not able to use this theorem to show that fragments S^i_2 are different assuming e.g. that Polynomial Hierarchy does not collapse. (This is however possible for the intuitionistic version of these fragments using the intuitionistic version of Theorem 1 which was proved in [B3].) Therefore we shall use more complex formulae than Σ^b_i, but then also we have to use a stronger mode of computation—this will be just the computations with counterexamples. The concept of counterexamples is also not new in proof theory, it goes back to Kreisel [K]. Recently Jan Krajíček [Kr] noted a close connection between the present concept and the former one, which can be used to give an alternative proof of the following theorems.

Theorem 2 [KPT]. *Suppose that for $i > 0$, and φ in Σ^b_{i+1}*

$$T^i_2 \vdash \forall x \exists y \forall z \leq x \varphi(x, y, z).$$

Then, for a given x, one can compute y such that $\forall z \leq x \varphi(x, y, z)$ in polynomial time using a Σ^p_i oracle by a type 0 *counterexample computation (i.e. using a constant number of counterexamples).*

The following is a new result.

Theorem 3. *Suppose that for $i > 0$, and φ in Σ^b_{i+1}*

$$S^{i+1}_2 \vdash \forall x \exists y \forall z \leq x \varphi(x, y, z).$$

Then, for a gien x, one can compute y such that $\forall z \leq x \varphi(x, y, z)$ in polynomial time using a Σ^p_i oracle by a type 1 *counterexample computation (i.e. using an unbounded number of counterexamples).*

Let C, ρ be an optimization problem with C in Π^p_i and ρ polynomial time computable. We have constructed a Σ^b_{i+1} formula $B(x, y, z)$ such that $\forall z \leq y B(x, y, z)$ expresses that y is an optimal solution to x. Thus we can apply Theorems 2 and 3.

Corollary 1. *For $i \geq 0$ and C in Π_i^p, if T_2^i proves that $C(x,y), \rho(y)$ has an optimal solution for every x, then optimal solutions can be computed using a type 0 computation with a Σ_i^p oracle.*

Corollary 2. *For $i \geq 0$ and C in Π_i^p, if S_2^{i+1} proves that $C(x,y), \rho(y)$ has an optimal solution for every x, then optimal solutions can be computed using a type 1 computation with a Σ_i^p oracle.*

By Proposition 1 we know that the existence of optimal solutions (for certain type of problems) is equivalent to (certain type of) induction. Thus we get:

Corollary 3 [KPT]. *For $i \geq 0$, if $T_2^i = S_2^{i+1}$, then every type 1 optimization problem C, ρ with C in Π_i^p can be computed by a type 0 computation with a Σ_i^p oracle.*

Corollary 4. *For $i \geq 0$, if $S_2^{i+1} = T_2^{i+1}$, then every optimization problem C, ρ with C in Π_i^p can be computed by a type 1 computation with a Σ_i^p oracle.*

Conclusions in both corollaries seem unlikely which strongly suggests that $T_2^i \neq S_2^{i+1}$ and $S_2^{i+1} \neq T_2^{i+1}$. In [KPT] it has been shown that the conclusion of Corollary 3 implies that $\Sigma_{i+2}^p = \Pi_{i+2}^p$ which is usually conjectured to be false. For the conclusion of Corollary 4 it is an open problem, whether it implies anything like that.

7. Proof of Theorem 3

Here we shall sketch the idea of a proof of Theorem 3. The proof is a modification of Buss' proof of Theorem 1. Theorem 2 was proved using different means. We shall observe that it can be obtained by modifying the proof given below. Hence there is a uniform way to prove all three theorems.

We consider the sequent calculus of Schwichtenberg [Sch]. The sequents are sets of formulae; logical connectives are $\&, \vee, \neg$, where negation is allowed only at atomic formulae (if φ is not atomic, $\neg\varphi$ is an abbreviation for the equivalent formula obtained by applying De Morgan's laws). The system has initial sequents of the form $\Gamma, \varphi, \neg\varphi$, (which is $\Gamma \cup \{\varphi, \neg\varphi\}$), a rule for $\&$, two rules for $\vee$, one rule for each quantifier and a cut rule. The rules which are important for the proof will be explained in the course of the proof. We formalize S_2^{i+1} in this system by allowing initial sequents of the form Γ, φ (which is $\Gamma \cup \{\varphi\}$) for φ a basic axiom and by adding the following rule for each $\psi(x)$ in Σ_{i+1}^b:

$$\frac{\Theta, \neg\psi(\lfloor b/2 \rfloor), \psi(b)}{\Theta, \neg\psi(0), \psi(t)},$$

where b is not free in Θ and t is a term.

Let $\varphi(a, y, z)$ in Σ_{i+1}^b be fixed, let $A(a)$ be defined by

$$A(a) \equiv \exists y \forall z \varphi(a, y, z).$$

In order to simplify notation we shall assume that the bound $z \leq a$ is implicit in $\varphi(a, y, z)$. Suppose a proof of $A(a)$ is given. By cut elimination we can assume that it is free-cut-free, i.e. the cut formulae are only substitution instances of basic axioms or induction formulae. Thus this proof contains only Σ_{i+1}^b and Π_{i+1}^b formulae and substitution instances of subformulae of $A(a)$, which are either $A(a)$ itself, or $\forall z \varphi(a, t, z)$, for some term t, or a Σ_{i+1}^b formulae. Thus the general form of a sequent Γ in the proof is

$$\Pi, \Sigma, \Delta, \Phi, A(a),$$

where

Π are Π_{i+1}^b formulae which are not in $\Pi_i^b \cup \Sigma_i^b$;
Σ are Σ_{i+1}^b formulae which are not in $\Pi_i^b \cup \Sigma_i^b$;
Δ are $\Pi_i^b \cup \Sigma_i^b$ formulae;
Φ is a set of formulae of the form $\forall x \varphi(a, t, z)$;

and we can assume that $A(a)$ is present in each sequent.

Now we are going to define the concept of *witnessing functions* for such a Γ. Recall that we have defined Σ_{i+1}^b (Π_{i+1}^b resp.) formulae so that they consist of a prefix of existential (universal resp.) bounded quantifiers followed by a Π_i^b (Σ_i^b resp.) formula. We shall call these quantifiers *essential.* Let $a, b_1, \ldots, b_k$ be the string of free variables of Γ. We shall denote strings of variables and functions by boldface letters (e.g. $\mathbf{b}$ denotes $b_1, \ldots, b_k$). We choose distinct variables $x_1, \ldots, x_l$ for all (distinct) occurrences of variables at essential quantifiers in Π and, similarly, $y_1, \ldots, y_m$ for variables at essential quantifiers in Σ. A string of functions

$$f_1(a, \mathbf{b}, \mathbf{x}), \ldots, f_m(a, \mathbf{b}, \mathbf{x}), g(a, \mathbf{b}, \mathbf{x}),$$

will be called *witnessing functions* for Γ if the following formula is true for all assignments of natural numbers to $a, \mathbf{b}, \mathbf{x}$:

$$(*) \qquad \bigwedge \neg\Pi(x) \to \bigvee \Sigma(\mathbf{f}(a,\mathbf{b},\mathbf{x})) \vee \bigvee \Delta \vee A(a) \vee \forall z \varphi(a, g(a,\mathbf{b},\mathbf{x}), z).$$

Here $\Pi(\mathbf{x})$ denotes Π with essential bounded quanfiers omitted an their variables replaced by $x_1, \dots, x_l$; similarly in $\Sigma(\mathbf{f}(a,\mathbf{b},\mathbf{x}))$ essential quantifiers are omitted and their variables are replaced by $f_1(a,\mathbf{b},\mathbf{x}), \dots, f_m(a,\mathbf{b},\mathbf{x})$. Note that no witnessing functions occur in Φ, though the formulae in Φ are not in Π^b_{i+1}.

Using induction on the depth of a sequent in the proof, we shall show that every sequent has witnessing functions computable in the following way. THere is a Turing machine with a Σ^b_i oracle which on an input $a, \mathbf{b}, \mathbf{x}$ produces the values $f_1(a,\mathbf{b},\mathbf{x}), \dots, f_m(a,\mathbf{b},\mathbf{x}), g(a,\mathbf{b},\mathbf{x})$ in polynomially many steps. During the computation it may ask queries of the form $\forall z \varphi(a,u,z)$? where u is some value produced during the computation. If the answer is negative, it gets a counterexample, i.e. some z_0 such that $\neg\varphi(a,u,z_0)$. If the answer is positive, it puts $g(a,\mathbf{b},\mathbf{x}) = u$ and some default values for f_i's and stops. In aprticular, we get the type of computation required in the theorem for the end sequent, since it consists of $A(a)$ only. As defined above, we have to consider all possible strategies for the person (*TEACHER*) who answers the queries. Thus it would be more appropriate to talk about *functionals* $f_1, \dots, f_m$, rather than functions. Or we can say: for any strategy of *TEACHER* the computed functions are witnessing functions for the sequent.

The induction steps are similar to those in Buss' proof, except for those rules where the principalformula is a subformula of $A(a)$.

(1) Consider the following instance of the $\forall$-rule:

$$\frac{\Theta, \varphi(a,t,b_h)}{\Theta, \forall z \varphi(a,t,z)}.$$

Suppose we have witnessing functions $\mathbf{f}(a,\mathbf{b},\mathbf{x}), g(a,\mathbf{b},\mathbf{x})$ for the upper sequent. The lower sequent has free variables $a, \mathbf{b}'$, where $\mathbf{b}' = (b_1, \dots, b_{h-1}, b_{h+1}, \dots, b_k)$. To get the witnessing functios for the lower sequent, we omit the witnessing functions for $\varphi(a,t,z)$ and change the remaining ones as follows. First we compute the value of the term t, then we ask the query "$\forall z \varphi(a,t,z)$?". If the answer is positive, then the lower sequent is witnessed no matter how we define the functions. If the answer is negative and the counterexample is u, then we define the witnessing functions for the lower sequent by substituting u for b_i. Thus the functions depend only on $a, \mathbf{b}', \mathbf{x}$.

(2) Suppose $\exists$-rule is applied to $\forall x\varphi(a,t,z)$ to obtain $\exists y\forall z\varphi(a,y,z)$. Since this is just $A(a)$, which is present in every sequent, the instance looks like this:

$$\frac{\Theta, \forall z\varphi(a,t,z)}{\Theta}.$$

Suppose we have $\mathbf{f}, g$ for the upper sequent. The computation of $\mathbf{f}', g'$ for the lower sequent will be the following. First compute the value of t, then ask "$forallz\varphi(a,t,z)$?". If the answer is positive, set $g'(a,\mathbf{b},\mathbf{x}) = t$ and the value of $\mathbf{f}'$ is irrelevant; otherwise compute $\mathbf{f}', g'$ as for the upper sequent.

(3) Consider an instance of the PIND Σ^b_{i+1} rule:

$$\frac{\Theta, \forall x_1 \neg\psi(\lfloor b_1/2 \rfloor, x_1), \exists y_1 \psi(b_1, y_1)}{\Theta, \forall x_1 \neg\psi(0, x_1), \exists y_1 \psi(t, y_1)}$$

where ψ is Π^b_i. In order to simplify notation we assume that there is only one essential bounded quantifier and we omit the bound; also we have chosen the indices to be equal to 1. Suppose we have witnessing functions $\mathbf{f}(a,\mathbf{b},\mathbf{x}), g(a,\mathbf{b},\mathbf{x})$ for the upper sequent. We shall assume that $\exists y_1$ is witnessed by f_1. Now we define witnessing functions $\mathbf{f}'(a,\mathbf{b}',\mathbf{x}), g'(a,\mathbf{b}',\mathbf{x})$ for the lower sequent, where $\mathbf{b}' = (b_2, \ldots, b_k)$. First we compute $0 = v_0, \ldots, v_r = t$ such that $\lfloor v_{j+1}/2 \rfloor = v_j$, for $j = 0, \ldots, r-1$. Then we compute $\mathbf{f}^{(s)}, g^{(s)}$ as follows. Set

$$f_1^{(0)} = x_1,$$

and for $s \geq 0$ let

$$f_j^{(s+1)} = f_j(a, v_s, \mathbf{b}', f_1^{(s)}, x_2, \ldots, x_l),$$
$$g^{(s+1)} = g(a, v_s, \mathbf{b}', f_1^{(s)}, x_2, \ldots, x_l),$$

($f_j^{(0)}$ is not defined for $j > 1$, $f_1^{(s)}$ are iterations of f_1). In each step of the iteration $s = 0, 1, \ldots$ we also check whether $\psi(v_s, f_1^{(s)})$ is true and whether Θ is witnessed by $\mathbf{f}^{(s)}, g^{(s)}$. The computation will stop if one of the following four cases occurs:

(i) we get a positive answer to a query "$\forall z\varphi(a,u,z)$?";
(ii) $\psi(0, f_1^{(0)})$ is not true, i.e. $\neg\psi(0, x_1)$ is true;
(iii) if Θ is witnessed by $\mathbf{f}^{(s)}, g^{(s)}$;
(iv) $s = r - 1$.

We define the witnessing functions according to which case occurs:

(i) we set $g'(a, \mathbf{b}, \mathbf{x}) = u$ and $A(a)$ is witnessed;

(ii) the lower sequent is witnessed independently of the values of $\mathbf{f}', g'$;

(iii) if Θ is witnessed by $\mathbf{f}^{(s)}, g^{(s)}$, then we take them as $\mathbf{f}', g'$;

(iv) define $\mathbf{f}', g'$ as $\mathbf{f}^{(r-1)}, g^{(r-1)}$.

We only have to check that the lower sequent is witnessed also in case (iv). If (iv) occurs, then none of the (i)–(iii) has occurred before, in particular $\psi(0, f_1^{(0)})$ is true. Suppose $\psi(v_r, f^{(r-1)})$ is false. Then there is some $s < r$ such that $\psi(v_s, f_1^{(s)})$ is true and $\neg\psi(v_{s+1}, f_1^{(s)})$ is false. Since $\mathbf{f}, g$ witness the upper squent, this is possible only if Θ is witnessed by $\mathbf{f}^{(s)}, g^{(s)}$, which is a contradiction. Thus the lower sequent is witnessed also in case (iv).

We have tacitly assumed that $\exists y_1 \psi(t, y_1)$ does not occur in Θ. If it does, then $f_1^{(s)}$ will not be included in $\mathbf{f}'$ and case (iv) will be subsumed in case (iii).

We hope that this illustrates sufficiently well the changes that must be done in Buss' proof, and we are not going to consider other instances of rules and axioms. To state the main idea briefly: the change is in the possibility that a positive answer to a query "$\forall z \varphi(a, t, z)$?" may occur. In such a case the computation stops, since we have a witness for $A(a)$. $\square$

Now we describe a proof of Theorem 2. The assumptions are similar, except that we have a weaker rule of induction

$$\frac{\Theta, \neg\psi(b_h), \psi(b_n + 1)}{\Theta, \neg\psi(0), \psi(t)}$$

since ψ is only Σ_i^b. We use the same definition of witnessing as in the above proof, hence *no witnessing functions occur in* $\neg\psi(b_h), \psi(b_h + 1), \neg\psi(0), \psi(t)$. Let $\mathbf{f}, g$ be witnessing functions for the upper sequent. We shall define witnessing functions $\mathbf{f}', g'$ for the lower sequent. Let $a, \mathbf{b}', \mathbf{x}$ be input, where again $\mathbf{b}' = (b_1, \dots, b_{h-1}, b_{h+1}, \dots, b_k)$. First we check whether $\neg\psi(0) \vee \psi(t)$ is true. If it is true, then we take arbitrary values for $\mathbf{f}', g'$. If not, then we use binary search to find some u such that $\psi(u)$ & $\neg\psi(u + 1)$, $u < t$. This is possible, since now we can use $\psi(x)$ as an oracle. Then we put

$$\mathbf{f}'(a, \mathbf{b}', \mathbf{x}) = \mathbf{f}(a, b_1, \dots, b_{h-1}, u, b_{h+1}, \dots, b_k, \mathbf{x}),$$
$$g'(a, \mathbf{b}', \mathbf{x}) = g(a, b_1, \dots, b_{h-1}, u, b_{h+1}, \dots, b_k, \mathbf{x}).$$

In this case we witness Θ. Here no iterations of witnessing functions occur. Hence it holds for *every* rule: if the number of queries is constant (i.e. does not depend on parameters $a, \mathbf{b}, \mathbf{x}$), then it is constant in the lower sequent too. Consequently the number of queries used to compute the witness for the end sequent is constant. (It is not hard to prove a more precise upper bound: the number of queries is bounded by the number of applications of $\forall$-rule to formulae $\varphi(a, t, b_h)$.) $\square$

7. Relativizations

We conjecture that there are optimization problems C, ρ, with C in Π_i^p (of type 2) whose optimal solutiosn cannot be computed in polynomial time using counterexample computations (of type 1) with Σ_i^p oracles. By Corollary 4, this conjecture implies that $S_2^{i+1} \neq T_2^{i+1}$. We shall justify this conjecture for $i = 0$ and $i = 1$ by showing that for suitable oracles the relativized version is true. Clearly it is sufficient to prove it for $i = 1$. These results imply separations of the corresponding fragments of the system obtained from S_2 by adding a new uninterpreted predicate, (see Corollary 6 below).

In the following proof it will be convenient to consider oracles as mappings $A : \mathbb{N}^3 \to \{0, 1\}$.

Theorem 4. *There exists an oracle A such that there is no polynomial time interactive algorithm (type 1) with a $(\Sigma_1^p)^A$ oracle which computes the largest y such that $y \leq x$ and*

$$\forall u \leq x(A(x, y, u) = 0) \vee y = 0.$$

I.e. in the optimization problem $C(x, y)$ is $\forall u \leq x(A(x, y, u) = 0) \vee y = 0$, which is in $(\Pi_1^p)^A$, and $\rho(y) = y$. In the proof of Theorem 5 we shall assume some familiarity with the concept of relativization. As usual we shall use finite approximations to oracle A. The key lemma which enables us to diagonalize at level Σ_1^p is the following.

Lemma 1. *Let $R^\alpha(v)$ be a $(\Sigma_1^p)^\alpha$ predicate, where α is a variable for an oracle. Let a partial mapping A' be given, let v be given. Then there exists an extension A'' of A' such that it has only polynomially more elements than A' (i.e. $|A'' \backslash A'| \leq q(|v|)$, for some polynomial q), and for any two extensions A and B of A''*

$$R^A(v) \equiv R^B(v),$$

i.e. A'' *forces* $R(v)$ *or* $\neg R(v)$.

Proof. First uppose that there exists some $A_0 \supseteq A'$ such that $R^{A_0}(v)$ holds true. Take an accepting computation for v which uses A_0 and add to A' the queries asked by this computation. Any extension which gives the same answer to these queries will allow this accepting computation. If there is no such extension A_0, then put $A'' = A'$. □

Proof of Theorem 5. We construct the oracle in ω steps. In the i-th step we diagonalize the i-th *STUDENT*, which is a polynomially bounded Turing machine and a Σ_1^p oracle. The precise meaning of this statement is that we construct a finite extension A_i of the previous approximation to the oracle, we take an input x_i and define a strategy for *TEACHER* so that for any extension of A_i, *STUDENT* does not compute the optimal solution to x_i in $p(|x_i|)$ steps, where p is the polynomial bound to *STUDENT*. We can always take x_i so large that on x_j with $j < i$, *STUDENT* never asks queries of the form "$A(x_i, r, s) = 0$?".

On input x, *STUDENT* uses Σ_1^p oracle only for inputs whose size is polynomially bounded in $|x|$. Hence there is a polynomial bound $q'(|x|)$ to all possible values $q(|v|)$ for queries v asked during the computation on input x, (q is the polynomial from Lemma 1). Put $p'(|x|) = p(|x|) \cdot q'(|x|)$, (we assume also $q'(|x|) \geq 1$). We choose x_i so large that

$$(p(|x_i|) + 1) \cdot (p'(|x_i|) + 1) < x_i.$$

Each A_i is also constructed in several stages,

$$A_i^0 = A_{i-1}, A_i^1, A_i^2, \dots .$$

These stages will correspond to the queries of *STUDENT*. At the same time we define the strategy for *TEACHER*. We have to define the strategy of *TEACHER* only for $x_1, x_2, \dots$, since other inputs are not used for the diagonalization. Let i be given. The strategy of *TEACHER* will consist of her answers $y_1, y_2, \dots$. The approximations A_i^k will be constructed using Lemma 1, hence if we take *any* extension of A_i^k and if *TEACHER* uses $y_1, \dots, y_{k-1}$, the computation of *STUDENT* will be the same up to the k-th query. The number of these stages is bounded by the number of queries that *STUDENT* can pose to *TEACHER* and this in turn is bounded by the total running time $p(|x_i|)$ of *STUDENT*.

There are three reasons to extend the current approximation to the oracle A:

(1) to force the computation of *STUDENT*;
(2) to force that the answers of *TEACHER* are correct;
(3) to force that *STUDENT* has asked the wrong question (this can be avoided by taking an enumeration of *STUDENTS* who do not ask wrong questions).

By Lemma 1, we can always add $q'(|x_i|)$ new elements into the approximation of the oracle so that an answer of Σ_1^p oracle is forced. In this way we add at most $q'(|x_i|)$ new elements to the approximation to A in each computation step. Hence for (1) we have to add at most $p'(|x_i|)$ elements. To ensure (3) we need just one element. Once *STUDENT* asked a wrong question (i.e. he asked *TEACHER* whether y is maximal such that $\forall z \leq x_i A_i(x_i, y, z) = 0$ for some y such that $\exists z \leq x_i A_i(x_i, y, z) = 1$), the construction of A_i and *TEACHER*'s strategy is finished. For (2) we have to add all triples (x_i, y_k, z) for each answer y_k of *TEACHER* and each $z \leq x$. We shall construct A_i^k and y_k in such a way that we add new elements into the domain only if we need them because of one of the reasons (1)–(3) and the following condition is satisfied:

> for $k > 0$, y_k is the largest answer of *TEACHER* and $y_k \leq k \cdot (p'(|x_i|) + 1)$.

Suppose the condition is satisfied at stage $k-1$. We take any extension B of A_i^{k-1} and let *STUDENT* work until he presents a conjecture to *TEACHER*. Let B_i^{k-1} be an approximation which forces this computation of *STUDENT*, $A_i^{k-1} \subseteq B_i^{k-1} \subseteq B$, and let the conjecture of *STUDENT* be y. If $y > y_{k-1}$, then we can extend B_i^{k-1} to A_i^k by adding just one element to it in such a way that *STUDENT*'s answer is wrong (for any extension of A_i^k). This is because, for such a $y \leq x_i$, there can be at most $p'(|x_i|) < x_i$ elements $z \leq x_i$ such that $B_i^{k-1}(x_i, y, z)$ is defined. Otherwise $y \leq y_{k-1}$. There are at most $p'(|x_i|)$ elements $y' > y_{k-1}$ such that $B_i^{k-1}(x_i, y', z)$ is defined. By the assumption that the condition holds for $k-1$,

$$\begin{aligned} y_{k-1} + p'(|x_i|) &\leq (k-1) \cdot (p'(|x_i|) + 1) + p'(|x_i|) \leq \\ &\leq k \cdot (p'(|x_i|) + 1) \leq p(|x_i|) \cdot (p'(|x_i|) + 1) < x_i. \end{aligned}$$

Hence we can take y_k such that $y_{k-1} < y_k \leq x_i$, $B_i^{k-1}(x_i, y_k, z)$ is undefined for all z and

$$y_k \leq y_{k-1} + p'(|x_i|) + 1.$$

By the assumption that the condition holds for $k-1$,

$$y_k \le (k-1)\cdot(p'(|x_i|)+1)+p'(|x_i|)+1 = k\cdot(p'(|x_i|)+1).$$

Thus we can extend B_i^{k-1} to A_i^k by putting

$$A_i^k(x_i, y_k, z) = 0 \text{ for every } z \le x_i,$$

and the condition will be preserved.

After polynomially many steps *STUDENT* must stop, but we can still extend the oracle so that there exists a larger counterexample, because

$$\begin{aligned} y_k + p'(|x_i|) &\le k\cdot(p'(|x_i|)+1)+p'(|x_i|) \le \\ &\le (k+1)\cdot(p'(|x_i|)+1) \le (p(|x_i|)+1)\cdot(p'(|x_i|)+1) < x_i. \end{aligned}$$

Hence he is not able to find the optimal solution. □

Remarks. (1) The proof above is essentially the same as for the similar result in [KPT]. (2) We have shown more for this oracle: for every *STUDENT* there exists an input x such that either he asks a wrong question on x or he uses the trivial strategy on x without success.

Let $S_2^i(\alpha)$ (resp. $T_2^i(\alpha)$) be S_2^i (resp. T_2^i) extended by adding a new predicate to the language and extending PIND (resp. IND) to $\Sigma_i^b(\alpha)$ formulae (which are defined as Σ_i^b in the extended language).

Corollary 5. *For $i=1$ and $i=2$, $T_2^i(\alpha) \ne S_2^i(\alpha)$.*

Proof. First it is necessary to check that Proposition 1, Theorem 3 and hence also Corollary 4 can be relativized by adding a new uninterpreted predicate α. Take $C(x,y)$ to be

$$\forall u \le x(\alpha(x,y,u)=0) \vee y=0,$$

and $\rho(y)=y$. Then, by relativized Corollary 4, for any interpretation of α as a subset $A \subseteq \mathbb{N}$, we should be able to compute y form x using an interactive computation with oracle A. If we choose the interpretation fo α to be A from Theorem 5, we get a contradiction. In this way, we obtain the result for $i=2$. For $i=1$ we take the same A and encode in it some $\mathcal{NP}^A$-complete problem. Thus former Σ_1^p sets become $\mathcal{P}$ sets. Or we can prove a theorem similar to Theorem 5 for this simpler case using a trivial modification of the proof above. □

8. Open Problems

We would like to prove the conjecture that there are optimization problems C, ρ with C in Π_i^p which cannot be computed by (type 1) interactive computations with Σ_i^p oracles, since it implies that $S_2^{i+1} \neq T_2^{i+1}$. As this conjecture implies that $\mathcal{P} \neq \mathcal{NP}$, it is hopeless to try to prove the conjecture directly. In the present situation the following two problems seem to be more feasible:

(1) Reduce the conjecture to the statement that Polynomial Hierarchy is proper, or a similar statement in complexity theory.
(2) Find oracles for each i such that the relativized statements are true.

A proof of (2) would imply $T_2^i(\alpha) \neq S_2^i(\alpha)$. Similar statements for type 1 optimization problems and type 0 interactive computations were proved in [KPT]. Note that there is a different approach to the separation of fragments of Bounded Arithmetic. It is based on proof systems for the propositional calculus [KP]. There we would need to show superpolynomial lower bounds to the length of proofs in certain proof systems for the propositional calculus. This is a weaker question than $\mathcal{NP} \neq co\mathcal{NP}$. Even less we know about the related problem:

(3) Is T_2^i partially conservative over S_2^i, e.g. is T_2^i $\forall\Pi_1^b$-conservative over S_2^i?

Some results on this problem have been recently obtained by Krajíček [Kr]. Also note taht Krajíček and Takeuti [KT] have constructed a consistency statement which is the strongest $\forall\Pi_1^b$-formula provable in T_2^i, hence it is the best candidate for a possible separation of T_2^i from S_2^i.

Acknowledgments

I would like to thank Jan Krajíček and Jiři Sgall for carefully reading the manuscript and suggesting several improvements.

References

[B1] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, Napoli, 1986.

[B2] ———, *Axiomatizations and conservation results for fragments of bounded arithmetic*, Contemporary Mathematics, AMS Proc. of Workshop in Logic and Computation, 1987 **106** (1990), 57–84.

[B3] ———, *The Polynomial Hierarchy and intuitionistic bounded arithmetic*, in "Structure in Complexity Theory", LNCS 223, Springer-Verlag, 1986, pp. 77–103.

[C] S. A. Cook, *Feasibly constructive proofs and the propositional calculus*, Proc. 7-th STOC (1975), 73–89.

[Kr] J. Krajíček, *No counterexample interpretation and interactive computations*, these proceedings.

[KP] J. Krajíček and P. Pudlák, *Quantified propositional calculi and fragments of bounded arithmetic*, Zeitschrift f. Math. Logik **36**(1) (1990), 29–46.

[KPT] J. Krajíček, P. Pudlák and G. Takeuti, *Bounded arithmetic and Polynomial Hierarchy*, Annals of Pure and Applied Logic, to appear.

[KT] J. Krajíček and G. Takeuti, *On induction-free provability*, Discrete Applied Mathematics (to appear).

[K] G. Kreisel, *On the interpretation of non-finitist proofs*, JSL 16 (1951), 241–267.

[PW] J. Paris and A. Wilkie, *On the scheme of induction for bounded arithmetic formulas*, Annals of Pure and Applied Logic **35**(3) (1987), 205–303.

[Pa] R. Parikh, *Existence and feasibility in arithmetic*, JSL 36 (1971), 494–508.

[Pu] P. Pudlák, *A note on bounded arithmetic*, Fundamenta Mathematicae, to appear.

[Sch] H. Schwichtenberg, *Proof Theory: Some applications of cut-elimination*, in "Handbook of Mathematical Logic", J. Barwise ed. (1977), 867–895.

[T] G. Takeuti, *Some relations among systems of bounded arithmetic*, Preprint.

Mathematical Institute ČSAV, Žitná 25, Praha 1, Czechoslovakia

Mathematical Sciences Research Institute, Berkeley CA 94720

A part of this manuscript was prepared while the author was a member of the Mathematical Sciences Research Institute, Berkeley.

Research at MSRI supported in part by NSF Grant DMS-8505550.