Theoretical Computer Science 288 (2002) 181–193

Theoretical Computer Science

www.elsevier.com/locate/tcs

On an optimal propositional proof system and the structure of easy subsets of TAUT

Zenon Sadowski

Institute of Mathematics, University of Białystok, 15-267 Białystok, ul. Akademicka 2, Poland

Abstract

In this paper we develop a connection between optimal propositional proof systems and structural complexity theory—specifically, there exists an optimal propositional proof system if and only if there is a suitable recursive presentation of the class of all easy (polynomial time recognizable) subsets of *TAUT*. As a corollary we obtain the result that if there does not exist an optimal propositional proof system, then for every theory T there exists an easy subset of *TAUT* which is not T-provably easy. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Classical propositional logic; Complexity of proofs; Complexity classes; Complexity of computation

1. Introduction

The first classification of propositional proof systems by their relative efficiency was done by Cook and Reckhow [4] in 1979. The key tool for comparing the relative strength of proof systems is p-simulation. Intuitively, a proof system h p-simulates a second one g if there is a polynomial time computable function translating proofs in g into proofs in h. A propositional proof system is called p-optimal if it p-simulates any propositional proof system. The question of the existence of a p-optimal propositional proof system and its nondeterministic counterpart, an optimal propositional proof system, posed by Krajíček and Pudlák [9], is still open.

It is not known whether many-one complete languages for **NP** $\cap$ **co**-**NP** and for **UP** exist. For these and other promise classes no recursively enumerable representation of appropriate sets of Turing machines is known. Moreover, Hartmanis and Hemachandra [5] and Kowalczyk [7] pointed out that **NP** $\cap$ **co**-**NP** and **UP** possess complete

E-mail address: sadowski@math.uwb.edu.pl (Z. Sadowski).

languages if and only if there are recursive enumerations of polynomial time clocked Turing machines covering languages from these classes.

In this paper we show that the question of the existence of optimal (p-optimal) propositional proof systems can be characterized in a similar manner. The main result of our paper shows that optimal proof systems for *TAUT* (the set of all propositional tautologies) exist if and only if there is a recursive enumeration of polynomial time clocked Turing machines covering all easy (recognizable in polynomial time) subsets of *TAUT*. This means that the problem of the existence of complete languages for promise classes and the problem of the existence of optimal proof systems for *TAUT*, although distant at first sight, are structurally similar. Since complete languages for promise classes have been unsuccesfully searched for in the past our equivalence gives some evidence of the fact that optimal propositional proof systems might not exist.

Our result can be related to the already existing line of research in computational complexity. After the revelation of the connection between the existence of optimal proof systems and the existence of many–one complete languages for promise classes in [12, 15], this subject has been intensively investigated. Köbler and Messner [8] formalized this relationship introducing the concept of test set, and showed that the existence of p-optimal proof systems for *TAUT* and for *SAT* (the set of all satisfiable boolean formulas) suffices to obtain a complete language for $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$. Messner and Torán showed in [12] that a complete language for **UP** exists in case there is a p-optimal proof system for *TAUT*. We believe that our results take the next step towards deeper understanding of this link between optimal proof systems and complete languages for promise classes.

The paper is organized as follows. In Section 2 we set down notation that will be used throughout the paper. Background information about propositional proof systems is presented in Section 3. The problems of the existence of complete languages for the classes $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ and **UP** and their characterization in terms of polynomial time clocked machines covering languages from these classes are presented in Section 4. In Section 5 we give a precise definition of a family of propositional formulas which will be used in the proofs of our main results. In Section 6 our main results are stated and proved. In the last section we discuss corollaries arising from the main results of the paper.

2. Preliminaries

We assume some familiarity with basic complexity theory, see [1]. The symbol Σ denotes, throughout the paper, a certain fixed finite alphabet. The set of all strings over Σ is denoted by $\Sigma^\star$. For a string x, $|x|$ denotes the length of x. For a language $A \subset \Sigma^\star$ the complement of A is the set of all strings that are not in A.

We use Turing machines (acceptors and transducers) as our basic computational model. We will not distinguish between a machine and its code. For a deterministic Turing machine M and an input w, $\mathit{TIME}(M;w)$ denotes the computing time of M on

w. When M is a nondeterministic Turing machine $\mathit{TIME}(M;w)$ is defined only for w's accepted by M and denotes the number of steps in the shortest accepting computation of M on w. For a Turing machine M the symbol $\mathrm{L}(M)$ denotes the language accepted by M. The output of a Turing transducer M on input $w \in \Sigma^\star$ is denoted by $M(w)$.

We consider deterministic and nondeterministic polynomial time clocked Turing machines with uniformly attached standard $n^k + k$ clocks which stop their computations in polynomial time (see [1]). We impose some restrictions on our encoding of these machines. From the code of any polynomial time clocked Turing machine we can detect easily (in polynomial time) the natural k such that $n^k + k$ is its polynomial time bound. Let $D_1, D_2, D_3, \dots$ and $N_1, N_2, N_3, \dots$ be, respectively, standard enumerations of all deterministic and nondeterministic polynomial time clocked Turing machines.

Recall that the classes **P**, **NP**, **co-NP** are, respectively, the class of all languages recognized by deterministic Turing machines working in polynomial time, the class of all languages accepted by nondeterministic Turing machines working in polynomial time and the class of complements of all languages from **NP**. The symbol *TAUT* denotes the set (of encodings) of all propositional tautologies over a fixed adequate set of connectives, *SAT* denotes the set of all satisfiable boolean formulas.

Finally, $\langle .,\dots,. \rangle$ denotes some standard polynomial time computable tupling function.

3. Propositional proof systems

The abstract notion of a propositional proof system was introduced by Cook and Reckhow [4] in the following way:

Definition 3.1. A propositional proof system is a function $f : \Sigma^\star \xrightarrow{onto} \mathit{TAUT}$ computable by a deterministic Turing machine in time bounded by a polynomial in the length of the input.

A string w such that $f(w) = \alpha$ we call a proof of a formula α.

A propositional proof system that allows short proofs to all tautologies is called a polynomially bounded propositional proof system.

Definition 3.2 (*Cook and Reckhow*). A propositional proof system is polynomially bounded if and only if there exists a polynomial $p(n)$ such that every tautology α has a proof of length no more than $p(|\alpha|)$ in this system.

The existence of a polynomially bounded propositional proof system is equivalent to one of the most fundamental problems in complexity theory.

Fact 3.3 (Cook and Reckhow). **NP** = **co-NP** *if and only if there exists a polynomially bounded propositional proof system.*

Cook and Reckhow were the first to propose a program of research aimed at attacking the **NP** versus **co-NP** problem by classifying propositional proof systems by their relative efficiency and then systematically studying more and more powerful concrete proof systems (see [2]). A natural way for such a classification is to introduce a partial order reflecting the relative strength of propositional proof systems. This was done in two different manners.

Definition 3.4 (*Cook and Reckhow*). Propositional proof system P polynomially simulates (p-simulates) propositional proof system Q if there exists a polynomial time computable function $f:\Sigma^\star \to \Sigma^\star$ such that for every w, if w is a proof of α in Q, then $f(w)$ is a proof of α in P.

Definition 3.5 (*Krajíček and Pudlák*). Propositional proof system P simulates propositional proof system Q if there exists a polynomial p such that for every tautology α, if α has a proof of length n in Q, then α has a proof of length $\leqslant p(n)$ in P.

Obviously, p-simulation is a stronger notion than simulation. We would like to pay attention to the fact that the simulation between proof systems may be treated as a counterpart of the complexity-theoretic notion of reducibility between problems. Analogously, the notion of a complete problem (a complete language) would correspond to the notion of an optimal proof system. The notion of an optimal propositional proof system was introduced by Krajíček and Pudlák [9] in two different versions.

Definition 3.6. A propositional proof system is optimal if it simulates every other propositional proof system.

A propositional proof system is p-optimal if it p-simulates every other propositional proof system.

The following open problem, posed by Krajíček and Pudlák, will be studied in our paper.

Problem 3.7. (1) Does there exist an optimal propositional proof system?

(2) Does there exist a p-optimal propositional proof system?

The importance of these questions and their connection with the **NP** versus **co-NP** problem is described by the following fact.

Fact 3.8. *If an optimal* (p*-optimal*) *propositional proof system exists, then* **NP** $=$ **co-NP** *if and only if this system is polynomially bounded.*

4. Complete languages for NP ∩ co-NP and for UP

The classes **NP** ∩ **co-NP** and **UP** are called promise classes because they are defined using nondeterministic polynomial time clocked Turing machines which obey special conditions (promises). The problem whether a given nondeterministic polynomial time clocked Turing machine indeed defines a language in any of these classes is undecidable and because of this complete languages for these classes are not known. Since there exist relativizations for which these two classes have complete languages as well as relativizations for which they do not the problems of the existence of complete languages for $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ and $\mathbf{UP}$ seem to be very difficult.

It turns out that the existence of complete languages for these classes depends on a certain structural condition on the set of machines defining languages from these classes. Since this condition is the chief motivation for our main theorems, we survey known results in this direction.

The class $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ is most often defined using complementary pairs of nondeterministic Turing machines. We will use strong nondeterministic Turing machines to define this class. A strong nondeterministic Turing machine is one that has three possible outcomes: "yes", "no" and "maybe". We say that such a machine accepts a language L if the following is true: if $x \in L$, then all computations end up with "yes" or "maybe" and at least one with "yes", if $x \notin L$, then all computations end up with "no" or "maybe" and at least one with "no".

If $N_1, N_2, N_3, \ldots$ is a standard enumeration of all nondeterministic polynomial time clocked Turing machines then

$$\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP} = \{L(N_i)\colon\ N_i \text{ is strong nondeterministic}\}.$$

The following theorem links the question of the existence of a complete language for $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ with the existence of a recursively enumerable list of machines covering languages from $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$. In [7] this list of machines is called a "nice" presentation of $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$.

Theorem 4.1 (Kowalczyk). *There exists a complete language for* $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ *if and only if there exists a recursively enumerable list of strong nondeterministic polynomial time clocked Turing machines* $N_{i_1}, N_{i_2}, N_{i_3}, \ldots$ *such that* $\{L(N_{i_k})\colon\ k \geqslant 1\} = \mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$.

This theorem can be exploited to obtain the following independence result. Let T be any formal theory whose language contains the language of arithmetic, i.e. the language $\{0, 1, \leqslant, =, +, \cdot\}$. We will not specify T in detail but only assume that T is sound (that is, in T we can prove only true theorems) and the set of all theorems of T is recursively enumerable.

Theorem 4.2 (Kowalczyk). *If* $\mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ *has no complete languages, then for any theory* T *there exists* $L \in \mathbf{NP} \cap \mathbf{co}\text{-}\mathbf{NP}$ *such that for no nondeterministic polynomial time clocked* N_i *with* $L(N_i) = L$ *can it be proven in* T *that* N_i *is strong nondeterministic.*

The class $\mathbf{UP}$ is closely related to a one-way function, the notion central to public-key cryptography (see [13]). This class can be defined using categorical (unambiguous) Turing machines. We call a nondeterministic Turing machine categorical or unambiguous if it has the following property: for any input x there is at most one accepting computation. We define $\mathbf{UP} = \{L(N_i)\colon\ N_i \text{ is categorical}\}$. As we can see from the following theorems the problem of the existence of a complete language for **UP** is similar to its $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ counterpart.

Theorem 4.3 (Hartmanis and Hemachandra). *There exists a complete language for* **UP** *if and only if there exists a recursively enumerable list of categorical nondeterministic polynomial time clocked Turing machines* $N_{i_1}, N_{i_2}, N_{i_3}, \ldots$ *such that* $\{L(N_{i_k})\colon\ k \geqslant 1\} = \mathbf{UP}$.

Theorem 4.4 (Hartmanis and Hemachandra). *If* **UP** *has no complete languages, then for any theory* T *there exists* $L \in \mathbf{UP}$ *such that for no nondeterministic polynomial time clocked* N_i *with* $L(N_i) = L$ *can it be proven in* T *that* N_i *is categorical.*

In Sections 6 and 7 we will show that the similarity between the problems of the existence of complete languages for $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ and for **UP** is also shared by the problem of the existence of an optimal propositional proof system.

5. Formulas expressing the soundness of Turing machines

In this section we construct boolean formulas which will be used to verify for a given deterministic polynomial time clocked transducer M and integer n that M on any input of length n produces propositional tautologies. We use these formulas in the proofs of Theorems 6.3 and 6.6.

For any transducer N we will denote by f_N the function computed by N ($f_N : \Sigma^\star \to \Sigma^\star$).

Definition 5.1. A Turing transducer N is called sound if f_N maps $\Sigma^\star$ into $TAUT$ ($f_N : \Sigma^\star \to TAUT$).

To any polynomial time clocked transducer M we will assign the set $A_M = \{Sound_M^1, Sound_M^2, Sound_M^3, \ldots\}$ of propositional formulas such that: $Sound_M^n$ is a propositional tautology if and only if for every input of length n, the machine M outputs a propositional tautology.

So, for any polynomial time clocked transducer M, it holds: M is sound if and only if $A_M \subset TAUT$.

Let N be a fixed nondeterministic Turing machine working in polynomial time which accepts a string w if and only if w is not a propositional tautology. For any fixed polynomial time clocked transducer M, let us consider the set $B_M = \{\langle M, 0^n \rangle\colon$ There exists a string x of length n such that $M(x) \notin TAUT\}$. Using the machines M and N we construct the nondeterministic Turing machine M' which guesses a string x of length n, runs M on input x and then runs N on output produced by M.

Clearly M' works in polynomial time and accepts B_M. Let $F_{M,n}$ be Cook's Theorem formula (see [3]) for the machine M' and the input $\langle M, 0^n\rangle$. We define $Sound_M^n$ as $\neg F_{M,n}$ and then the formula $Sound_M^n$ is a tautology if and only if for every input of length n, M outputs a tautology. From the structure of Cook's reduction (as $F_{M,n}$ clearly displays M and n) it follows that for any fixed M, the set A_M is in **P**.

Moreover, the formulas describing the soundness of Turing machines possess the following properties:

(1) *Global uniformity property*: There exists a polynomial time computable function f such that for any polynomial time clocked transducer N with time bound $n^k + k$ and for any $w \in \Sigma^\star$

$$f(\langle N, w, 0^{|w|^k+k}\rangle) = Sound_N^{|w|}.$$

(2) *Local uniformity property*: Let M be any fixed polynomial time clocked transducer. There exists a polynomial time computable function f_M such that for any $w \in \Sigma^\star$

$$f_M(w) = Sound_M^{|w|}.$$

6. Main results

A class of sets is recursively presentable if there exists an effective enumeration of devices for recognizing all and only members of this class [10]. In this paper we use the notions of recursive **P**-presentation and recursive **NP**-presentation which are mutations of the notion of recursive presentability.

Definition 6.1. By an easy subset of $TAUT$ we mean a set A such that $A \subset TAUT$ and $A \in \mathbf{P}$ (A is polynomial time recognizable).

Definition 6.2. An optimal nondeterministic algorithm for $TAUT$ is a nondeterministic Turing machine M which accepts $TAUT$ and such that for every nondeterministic Turing machine M' which accepts $TAUT$ there exists a polynomial p such that for every tautology α

$$TIME(M; \alpha) \leqslant p(|\alpha|, TIME(M'; \alpha)).$$

Let A be any easy subset of $TAUT$. We say that nondeterministic polynomial time clocked Turing machine M names the set A if $L(M) = A$. Obviously, A may possess many names. The following theorem states that an optimal propositional proof system exists if and only if there exists a recursively enumerable list of names for all easy subsets of $TAUT$. We would like to pay attention to the similarity between the next theorem and Theorems 4.1 and 4.3 from Section 4.

Theorem 6.3. *Statements* (i)–(iii) *are equivalent*:

(i) *There exists an optimal propositional proof system.*

(ii) *There exists an optimal nondeterministic algorithm for TAUT.*

(iii) *The class of all easy subsets of TAUT possesses a recursive* **NP**-*presentation.*

By statement (iii) we mean: there exists a recursively enumerable list of nondeterministic polynomial time clocked Turing machines $N_{i_1}, N_{i_2}, N_{i_3}, \ldots$ such that

(1) $L(N_{i_j}) \subset TAUT$ for every j;

(2) for every $A \subset TAUT$ such that $A \in \mathbf{P}$ there exists j such that $A = L(N_{i_j})$.

Proof. (i) $\rightarrow$ (ii): With every propositional proof system we can associate a nondeterministic "guess and verify" algorithm for $TAUT$. On an input α this algorithm guesses a string w and then checks in polynomial time whether w is a proof of α. If successful, the algorithm halts in an accepting state.

Symmetrically, any nondeterministic algorithm for $TAUT$ can be transformed to a propositional proof system. The proof of a formula α in this system is a computation of M accepting α.

Let Opt denote an optimal propositional proof system and let M denote a nondeterministic Turing machine associated with Opt (a "guess and verify" algorithm associated with Opt). It can be easily checked that M accepts $TAUT$ and for any nondeterministic Turing machine M' accepting $TAUT$ there exists a polynomial p such that for every tautology α it holds:

$$TIME(M;\alpha) \leqslant p(|\alpha|, TIME(M';\alpha)).$$

(ii) $\rightarrow$ (iii): Let M be an optimal nondeterministic algorithm for $TAUT$. A recursive **NP**-presentation of all easy subsets of $TAUT$ we will define in two steps. In the first step we define a recursively enumerable list of nondeterministic Turing machines $F_1, F_2, F_3, \ldots$. The machine F_k is obtained by attaching the shut-off clock $n^k + k$ to the machine M. On any input w, the machine F_k accepts w if and only if M accepts w in no more than $n^k + k$ steps, where $n = |w|$. The sequence $F_1, F_2, F_3, F_4, \ldots$ of nondeterministic Turing machines possesses properties (1) and (2):

(1) for every i it holds $L(F_i) \subset TAUT$;

(2) for every A which is an easy subset of $TAUT$ there exists j such that $A \subset L(F_j)$.

To prove (2) let us consider A, an easy subset of $TAUT$ recognized by a Turing machine M'' working in polynomial time. Combining this machine with the "brute force" algorithm for $TAUT$ we obtain a deterministic (therefore also nondeterministic) Turing machine M' accepting $TAUT$. From the definition of M' it follows that there exists a polynomial p such that for any $\alpha \in A$, $TIME(M';\alpha) \leqslant p(|\alpha|)$. Since M is an optimal nondeterministic algorithm for $TAUT$ there exists a polynomial q such that for any $\alpha \in A$,
$TIME(M;\alpha) \leqslant q(|\alpha|)$. From this we conclude that for a sufficiently large j,
$TIME(M;\alpha) \leqslant |\alpha|^j + j$ for any $\alpha \in A$, hence $A \subset L(F_j)$.

In the second step we define the new recursively enumerable list of nondeterministic polynomial time clocked Turing machines $K_1, K_2, K_3, \ldots$. We define K_n, $n = \langle i, j \rangle$, as the machine which simulates $n^i + i$ steps of F_i and $n^j + j$ steps of N_j (see Section 2 for definition of N_j) and accepts w if and only if both F_i and N_j accept w.

Let A be any fixed easy subset of $TAUT$. There exist k and m such that $A = L(N_k)$ and $A \subset L(F_m)$. It follows from the definition of the sequence $K_1, K_2, K_3, \ldots$ that A is accepted by the machine $K_{\langle m,k \rangle}$, so $K_1, K_2, K_3, \ldots$ provides a recursive **NP**-presentation of all easy subsets of $TAUT$.

(iii) $\rightarrow$ (i): Let G be the machine generating the codes of the machines from the sequence $N_{i_1}, N_{i_2}, N_{i_3}, \ldots$ forming a recursive **NP**-presentation of all easy subsets of $TAUT$. We say that a string $v \in \Sigma^\star$ is in good form if

$$v = \langle M, w, Comp\text{-}G, Comp\text{-}Sound_M^{|w|}, 0^{|w|^k + k} \rangle,$$

where M is a polynomial time clocked Turing transducer with $n^k + k$ time bound, $w \in \Sigma^\star$, $Comp\text{-}G$ is a computation of the machine G. This computation produces a code of a certain machine N_{i_j}, $Comp\text{-}Sound_M^{|w|}$ is a computation of the machine N_{i_j} accepting the formula $Sound_M^{|w|}$, $0^{|w|^k + k}$ is the sequence of zeros (padding).

We call a Turing transducer n-sound if and only if on any input of length n it produces a propositional tautology.

Let us notice, that if v is in good form then $Sound_M^{|w|}$ as a formula accepted by a certain machine from **NP**-presentation is a propositional tautology. This clearly forces M to be n-sound, where $n = |w|$, so M on input w produces a propositional tautology.

Let α_0 be a certain fixed propositional tautology. We define $Opt : \Sigma^\star \rightarrow \Sigma^\star$ in the following way: $Opt(v) = \alpha$ if v is in good form

$$(v = \langle M, w, Comp\text{-}G, Comp\text{-}Sound_M^{|w|}, 0^{|w|^k + k} \rangle)$$

and α is a propositional tautology produced by M on input w, otherwise $Opt(v) = \alpha_0$. Clearly, $Opt : \Sigma^\star \overset{onto}{\rightarrow} TAUT$.

In order to prove that Opt is polynomial time computable it is sufficient to notice that using global uniformity property we can check in polynomial time whether v is in good form. Hence Opt is a propositional proof system.

It remains to prove that Opt simulates any propositional proof system. Let h be a propositional proof system computed by the polynomial time clocked transducer K with time bound $n^l + l$. Since the set $A_K = \{Sound_K^1, Sound_K^2, Sound_K^3, \ldots\}$ is an easy subset of $TAUT$, there exists the machine N_{i_j} from the **NP**-presentation such that $A_K = L(N_{i_j})$.

Let α be any propositional tautology and let x be its proof in h. Then α possesses a proof in Opt of the form:

$$\langle K, x, Comp\text{-}G, Comp\text{-}Sound_K^{|x|}, 0^{|x|^l + l} \rangle.$$

The word $Comp\text{-}G$ is the computation of G producing the code of N_{i_j}, $Comp\text{-}Sound_K^{|x|}$ is a computation of N_{i_j} accepting $Sound_K^{|x|}$. Let us notice that $|Comp\text{-}G| = c_1$, where c_1 is a constant. Because N_{i_j} is polynomial time clocked there exists a polynomial p such that $|Comp\text{-}Sound_K^{|x|}| \leqslant p(|x|)$. The constants c_1, l and the polynomial p depend only on N_{i_j} which is fixed and connected with K. This proves that Opt simulates h. $\square$

The following definition is a nondeterministic counterpart of Definition 6.1.

Definition 6.4. By an **NP**-easy subset of $TAUT$ we mean a set A such that $A \subset TAUT$ and $A \in \mathbf{NP}$.

A slight change in the previous proof shows that also the second version of Theorem 6.3 is valid. In this version condition (iii) is replaced by the following one:

(iv) The class of all **NP**-easy subsets of $TAUT$ possesses a recursive **NP**-presentation.

Now we will translate the previous result to the deterministic case.

Definition 6.5. An almost optimal deterministic algorithm for $TAUT$ is a deterministic Turing machine M which accepts $TAUT$ and such that for every deterministic Turing machine M' which accepts $TAUT$ there exists a polynomial p such, that for every tautology α

$$TIME(M;\alpha) \leqslant p(|\alpha|, TIME(M';\alpha)).$$

We name such an algorithm as an almost optimal deterministic algorithm for $TAUT$ because the optimality property is stated for any input string x which belongs to $TAUT$ and nothing is claimed for other x's (compare the definition of an optimal acceptor for $TAUT$ in [11]).

Equivalence (i)$\leftrightarrow$(ii) in the next theorem is restated from [9] in order to emphasize the symmetry between Theorems 6.3 and 6.6.

Theorem 6.6. *Statements* (i)–(iii) *are equivalent*:

(i) *There exists a p-optimal propositional proof system.*
(ii) *There exists an almost optimal deterministic algorithm for TAUT.*
(iii) *The class of all easy subsets of TAUT possesses a recursive* **P**-*presentation.*

By statement (iii) we mean: there exists a recursively enumerable list of deterministic polynomial time clocked Turing machines $D_{i_1}, D_{i_2}, D_{i_3}, \ldots$ such that
(1) $L(D_{i_j}) \subset TAUT$ for every j;
(2) for every $A \subset TAUT$ such that $A \in \mathbf{P}$ there exists j such that $A = L(D_{i_j})$.

Proof. (i)$\rightarrow$(ii): See [9].

(ii)$\rightarrow$(iii): This follows by the same arguments as in the proof of (ii)$\rightarrow$(iii) from Theorem 6.3. The only change is the use of deterministic Turing machines instead of the nondeterministic ones.

(ii)$\rightarrow$(iii): A string $v \in \Sigma^\star$ is in good form if

$$v = \langle M, w, Comp\text{-}G, Comp\text{-}Sound_M^{|w|}, 0^{|w|^k + k} \rangle,$$

where the appropriate symbols mean the same as before. We define $Opt : \Sigma^\star \to \Sigma^\star$ analogously as in the proof of Theorem 6.3: $Opt(v) = \alpha$ if v is in good form

$$(v = \langle M, w, Comp\text{-}G, Comp\text{-}Sound_M^{|w|}, 0^{|w|^k+k}\rangle)$$

and α is a propositional tautology produced by M on input w, otherwise $Opt(v) = \alpha_0$, where α_0 is a certain fixed propositional tautology.

It remains to prove that Opt p-simulates any propositional proof system. Let h be a propositional proof system computed by a polynomial time clocked transducer K with time bound $n^l + l$. Since the set $A_K = \{Sound_K^1, Sound_K^2, Sound_K^3, \ldots\}$ is an easy subset of $TAUT$, there exists the machine D_{i_j} from the **P**-presentation such that $A_K = L(D_{i_j})$. The function $t : \Sigma^\star \to \Sigma^\star$

$$t(x) = \langle K, x, Comp\text{-}G, Comp\text{-}Sound_K^{|x|}, 0^{|x|^l+l}\rangle.$$

translates proofs in h into proofs in Opt. The word $Comp\text{-}G$ in the definition of t is the computation of G producing the code of D_{i_j}, $Comp\text{-}Sound_K^{|x|}$ is a computation of D_{i_j} accepting $Sound_K^{|x|}$.

From the fact that D_{i_j} is deterministic and works in polynomial time and from local uniformity property (see Section 5) it follows that $Comp\text{-}Sound_K^{|x|}$ can be constructed in polynomial time. This proves that t is polynomial time computable. $\square$

Definition 6.7. A Turing machine acceptor M is called sound if and only if $L(M) \subset TAUT$.

The question, whether the set of all sound deterministic (nondeterministic) polynomial time clocked Turing machines yields the desired **P**-presentation (**NP**-presentation) (that is, whether this set is recursively enumerable) occurs naturally in connection with Theorems 6.3 and 6.6. The negative answer to this question is provided by the next theorem.

Theorem 6.8. *The set of all sound deterministic* (*nondeterministic*) *polynomial time clocked Turing acceptors is not recursively enumerable.*

This follows immediately from Rice's Theorem (see [14]).

7. Independence results

Let T be any formal theory satisfying the assumptions from Section 4. The notation $T \vdash \beta$ means that a first order formula β is provable in T.

Let M be a Turing machine.
By "$L(M) \subset TAUT$" we denote the first order formula which expresses the soundness of M, i.e. $\forall_{w \in L(M)}$ [w is a propositional tautology].

Definition 7.1. A deterministic (nondeterministic) Turing machine M is T-sound if $T \vdash$ "$L(M) \subset TAUT$".

Definition 7.2. A set $A \subset TAUT$ is T-provably **NP**-easy if there exists a nondeterministic polynomial time clocked Turing machine M fulfilling (1)–(2):
(1) M is T-sound;
(2) $L(M) = A$.

As in the case of the classes **NP** ∩ **co-NP** and **UP** we can obtain the following independence result.

Theorem 7.3. *If there does not exist an optimal propositional proof system, then for every theory T there exists an easy subset of $TAUT$ which is not T-provably* **NP**-*easy.*

Proof. Suppose, on the contrary, that there exists a theory T such that all easy subsets of $TAUT$ are T-provably **NP**-easy. Then the following recursively enumerable set of machines $\Omega_T = \{M$: M is a nondeterministic polynomial time clocked Turing machine which is T-sound$\}$ creates a recursive **NP**-presentation of the class of all easy subsets of $TAUT$. By Theorem 6.3, this implies that there exists an optimal propositional proof system, giving a contradiction. □

The following result can be obtained from the second version of Theorem 6.3.

Theorem 7.4. *If there does not exist an optimal propositional proof system, then for every theory T there exists an* **NP**-*easy subset of $TAUT$ which is not T-provably* **NP**-*easy.*

The translation of this result to the deterministic case goes along the following lines.

Definition 7.5. A set $A \subset TAUT$ is T-provably easy if there exists a deterministic polynomial time clocked Turing machine M fulfilling (1)–(2):
(1) M is T-sound;
(2) $L(M) = A$.

Theorem 7.6. *If there does not exist a p-optimal propositional proof system, then for every theory T there exists an easy subset of $TAUT$ which is not T-provably easy.*

8. Conclusion

In this paper we related the question of the existence of an optimal propositional proof system to the recursive presentability of the family of all easy subsets of $TAUT$ by means of polynomial time clocked Turing machines. The problems of the existence of complete languages for the classes **NP** $\cap$ **co-NP** and for **UP** have a similar characterization. From this characterization a variety of interesting results about the promise classes **NP** $\cap$ **co-NP** and **UP** were derived by recursion-theoretic techniques (see [5, 7]). Although recursion-theoretic methods seem unable to solve the problem of the existence of an optimal propositional proof system, we believe that our main results from Section 6 allow the application of these methods (as it was in case of promise classes, see [5, 6]) to further study of this problem.

References

[1] J. Balcazar, J. Díaz, J. Gabarró, Structural Complexity I, Springer, Berlin, 1995.

[2] S. Buss, Lectures on proof theory. Tech. Report No. SOCS-96.1, McGill University, 1966.

[3] S. Cook, The complexity of theorem proving procedures, Proc. 3rd ACM Symp. on Theory of Computing, 1971, pp. 151–158.

[4] S. Cook, R. Reckhow, The relative efficiency of propositional proof systems, J. Symbolic Logic 44 (1979) 36–50.

[5] J. Hartmanis, L. Hemachandra, Complexity classes without machines: on complete languages for UP, Theoret. Comput. Sci. 58 (1988) 129–142.

[6] J. Hartmanis, N. Immerman, On complete problems for NP $\cap$ co-NP, in: Proc. 12th Internat. Colloq. on Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 194, Springer, Berlin, 1985, pp. 250–259.

[7] W. Kowalczyk, Some connections between presentability of complexity classes and the power of formal systems of reasoning, in: Proc. Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, vol. 176, Springer, Berlin, 1988, pp. 364–369.

[8] J. Köbler, J. Messner, Complete problems for promise classes by optimal proof systems for test sets, in: Proc. 13th Ann. IEEE Conf. on Computational Complexity, IEEE Computer Soc. Press, Silver Spring, MD, 1998, pp. 132–140.

[9] J. Krajíček, P. Pudlák, Propositional proof systems, the consistency of first order theories and the complexity of computations, J. Symbolic Logic 54 (1989) 1063–1079.

[10] L. Landweber, R. Lipton, E. Robertson, On the structure of sets in NP and other complexity classes, Theoret. Comput. Sci. 15 (1981) 181–200.

[11] J. Messner, On optimal algoritms and optimal proof systems, in: Proc. 16th Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 1563, Springer, Berlin, 1999.

[12] J. Messner, J. Torán, Optimal proof systems for propositional logic and complete sets, in: Proc. 15th Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 1373, Springer, Berlin, 1998, pp. 477–487.

[13] C. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.

[14] H. Rice, Classes of recursively enumerable sets and their decision problems, Trans. Amer. Math. Soc. 74 (1953) 358–366.

[15] Z. Sadowski, On an optimal quantified propositional proof system and a complete language for NP $\cap$ co-NP, in: Proc. 11th Internat. Symp. on Fundamentals of Computing Theory, Lecture Notes in Computer Science, vol. 1279, Springer, Berlin, 1997, pp. 423–428.