On a p-optimal proof system for the set of all satisfiable boolean formulas (SAT)

Zenon Sadowski

Institute of Mathematics,
University of Białystok
15-267 Białystok, ul. Akademicka 2, Poland
e-mail: sadowski@math.uwb.edu.pl

Abstract. In this paper we show that the problem of the existence of a p-optimal proof system for SAT can be characterized in a similar manner as J. Hartmanis and L. Hemachandra characterized the problem of the existence of complete languages for UP. Namely, there exists a p-optimal proof system for SAT if and only if there is a suitable recursive presentation of the class of all easy (polynomial time recognizable) subsets of SAT. Using this characterization we prove that if there does not exist a p-optimal proof system for SAT, then for every theory T there exists an easy subset of SAT which is not T-provably easy.

1 Introduction

The question of the existence of a p-optimal proof system for SAT (the set of all satisfiable boolean formulas) was posed by J. Köbler and J. Messner in [6]. This question, like the similar and older problem of the existence of a p-optimal proof system for $TAUT$ (the set of all tautologies in Propositional Logic), is still open. The related problem of whether the natural proof system for SAT (a satisfying truth assignment is a proof of formula α) is p-optimal, was considered in different nomenclature by S. Fenner, L. Fortnow, A. Naik and J. Rogers in [3] (see also [7]).

A proof system for a language L is a polynomial time computable function whose range is L. This notion was defined by S. Cook and R. Reckhow in [2]. To compare the efficiency of different proof systems, they considered the notion of p-simulation. Intuitively a proof system h p-simulates a second one g if there is a polynomial time computable function t translating proofs in g into proofs in h. A proof system is called p-optimal for L when it p-simulates any proof system for L.

The existence of a p-optimal proof system for $TAUT$, historically the first question about optimal proof systems, was posed by J. Krajíček and P. Pudlák [8] in 1989. Since then, research concerning optimal proof systems has acquired importance in Proof Complexity Theory and Structural Complexity Theory. J. Köbler and J. Messner [6] proved that the existence of both p-optimal proof systems for SAT and for $TAUT$ suffices to obtain a complete language for

NP $\cap$ **co-NP**. This result generated interest in the problem of the existence of a p-optimal proof system for SAT .

It is not currently known whether **UP** and other promise classes have complete languages. J. Hartmanis, and L. Hemachandra pointed out in [4] that **UP** possesses complete languages if and only if there is a recursive enumeration of polynomial time clocked Turing machines covering all languages from this class. Earlier, the question of whether **NP** $\cap$ **co-NP** possesses complete languages was related to an analogous statement (see [5]).

In this paper we show that the question of the existence of a p-optimal proof system for SAT can be characterized in a similar manner. We prove that a p-optimal proof system for SAT exists if and only if there is a recursive enumeration of polynomial time clocked Turing machines covering all easy (polynomial time recognizable) subsets of SAT. This means that the problems of the existence of complete languages for promise classes and the problem of the existence of a p-optimal proof system for SAT, though 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 p-optimal proof systems for SAT might not exist.

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. The symbol FP denotes the class of functions that can be computed in polynomial time.

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, TIME(M, w) denotes the computing time of M on w.

We consider 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.

The symbol $TAUT$ denotes the set (of all encodings) of propositional tautologies; SAT denotes the set of all satisfiable boolean formulas.

Finally, $< .,\ldots,. >$ denotes some standard polynomial time computable tupling function.

3 P-optimal proof systems and almost optimal algorithms for SAT

The systematic study of the efficiency of propositional proof systems (proof systems for $TAUT$) was started by S. Cook and R. Reckhow in [2]. They introduced the abstract notion of a proof system for $TAUT$ and proved that a polynomially bounded proof system for $TAUT$ exists if and only if **NP=co-NP** (see [2, 10]).

The notion of a proof system can be considered not only for $TAUT$ but also for any language L.

Definition 1. *(see [2]) A proof system for L is a polynomial time computable function $h : \Sigma^\star \stackrel{onto}{\longrightarrow} L$.*

If $h(w) = x$ we may say that w is a proof of x in h. It follows from the above definition that for any proof system h there exists a polynomial time clocked transducer M_h which computes h.

To classify propositional proof systems by their relative efficiency S. Cook and R. Reckhow introduced in [2] the notion of p-simulation. For proof systems for SAT this notion can be defined as follows:

Definition 2. *(see [2]) Let h, h' be two proof systems for SAT. We say that h p-simulates h' if there exists a polynomial time computable function $\gamma : \Sigma^\star \longrightarrow \Sigma^\star$ such that for every $\alpha \in SAT$ and every $w \in \Sigma^\star$, if w is a proof of α in h', then $\gamma(w)$ is a proof of α in h.*

Definition 3. *(see [8]) A proof system for SAT is p-optimal if it p-simulates any proof system for SAT.*

The notion of an almost optimal deterministic algorithm for $TAUT$ was introduced by J. Krajíček and P. Pudlák [8]. We will consider an analogous notion of an almost optimal deterministic algorithm for SAT with Krajíček-Pudlák's optimality property.

Definition 4. *(cf. [11]) An almost optimal deterministic algorithm for SAT is a deterministic Turing machine M which recognizes SAT and such that for any deterministic Turing machine M' which recognizes SAT there exists a polynomial p such that for every satisfiable boolean formula α*

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

We name this algorithm as an almost optimal because the optimality condition is stated only for any input string x which belongs to SAT and nothing is claimed for other $x's$ (compare the definition of an optimal acceptor for SAT in [9]).

The following result was proved in [11] (see also [9]).

Theorem 1. *Statements (i) - (ii) are equivalent.*

(i) There exists an almost optimal deterministic algorithm for SAT.
(ii) There exists a p-optimal proof system for SAT.

4 P-optimal proof system for SAT and the structure of easy subsets of SAT

The problems of the existence of complete languages for **NP** $\cap$ **co-NP** and **UP** are oracle dependent (see [12] and [4]), and therefore they seem to be very difficult. It was shown in [5] and [4] that these problems can be related to statements about the existence of recursive enumerations of polynomial time clocked Turing machines covering all languages from these classes. We will show in this section that the problem of the existence of a p-optimal proof system for SAT can also be related to an analogous statement.

Definition 5. *We say that a polynomial time clocked transducer M behaves well on input w if M on w outputs a satisfiable boolean formula.*

To any polynomial time clocked transducer M and any $w \in \Sigma^\star$ we shall assign the boolean formula $TEST_{M,w}$ such that:
The formula $TEST_{M,w}$ is satisfiable if and only if M behaves well on input w.

Our construction of the formula $TEST_{M,w}$ is adapted from Cook's proof that SAT is **NP**-complete. Let S be a fixed nondeterministic Turing machine working in polynomial time and accepting SAT. Let M' be the Turing machine which on any input x runs M and then runs S on the output produced by M. The formula $TEST_{M,w}$ is just Cook's formula for the pair $< M', w >$.

Test formulas possess the following properties:

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

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

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

$$f_M(w) = TEST_{M,w}$$

Definition 6. *By an easy subset of SAT we mean a set B such that $B \subset SAT$ and $B \in$ **P**.*

Let $D_1, D_2, D_3, ...$ denote the standard enumeration of all polynomial time clocked deterministic Turing machines.

Theorem 2. *Statements (i) - (ii) are equivalent.*

(i) There exists a p-optimal proof system for SAT.
(ii) The class of all easy subsets of SAT possesses a recursive **P***-presentation.*

By the statement (ii) we mean: there exists a recursively enumerable list of deterministic polynomial time clocked Turing machines $D_{i_1}, D_{i_2}, D_{i_3}, ...$ such that

(1) $L(D_{i_j}) \subset SAT$ for every j
(2) For every $A \subset SAT$ such that $A \in \mathbf{P}$ there exists j such that $A = L(D_{i_j})$

Proof. $(i) \rightarrow (ii)$ The existence of a p-optimal proof system for SAT is equivalent to the existence of an almost optimal deterministic algorithm for SAT. Let M be such an algorithm. A recursive **P**-presentation of all easy subsets of SAT we will define in two steps. In the first step we define a recursively enumerable list of deterministic Turing machines F_1, F_2, F_3,... 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, ... of deterministic Turing machines possesses the properties (1) and (2):

(1) For every i it holds $L(F_i) \subset SAT$
(2) For every A which is an easy subset of SAT there exists j such that $A \subset L(F_j)$

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

In the second step we define the new recursively enumerable list of deterministic polynomial time clocked Turing machines K_1, K_2, K_3,... 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 D_j and accepts w if and only if both F_i and D_j accept w and rejects w in the opposite case.

Let A be any fixed easy subset of SAT. There exist k and m such that $A = L(D_k)$ and $A \subset L(F_m)$, hence A is recognized by the machine $K_{\langle m,k \rangle}$. From this we conclude that K_1, K_2, K_3, ... provides a recursive **P**-presentation of all easy subsets of SAT.

$(ii) \rightarrow (i)$

Let G be the machine generating the codes of the machines from the sequence D_{i_1}, D_{i_2}, D_{i_3},... forming a recursive **P**-presentation of all easy subsets of SAT. We say that a string $v \in \Sigma^\star$ is in good form if

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

where:

M is a polynomial time clocked Turing transducer with $n^k + k$ time bound, $w \in \Sigma^\star$,
$Comp - G$ is a computation of the machine G. This computation produces a code of a certain machine D_{i_j},

$Comp{-}TEST_{M,w}$ is a computation of the machine D_{i_j} accepting the formula $TEST_{M,w}$,
$0^{|w|^k+k}$ is the sequence of zeros (padding).

Let us note, that if v is in good form then $TEST_{M,w}$ is satisfiable as a formula accepted by a certain machine from **P**-presentation. This clearly forces M to behave well on input w, so M on input w produces a satisfiable boolean formula.

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

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

and α is a satisfiable boolean formula produced by M on input w, otherwise $Opt(v) = \alpha_0$.

Clearly, $Opt : \Sigma^\star \overset{onto}{\longrightarrow} SAT$.

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 proof system for SAT.

It remains to be proved that Opt p-simulates any proof system for SAT. Let h be a proof system for SAT computed by a polynomial time clocked transducer K with time bound $n^l + l$. From the structure of Cook's reduction (as $TEST_{K,w}$ clearly displays K and w) it follows that the set $A_K = \{TEST_{K,w} \colon w \in \Sigma^\star\}$ is an easy subset of SAT. Therefore there exists the machine D_{i_j} from the **P**-presentation such that $A_K = L(D_{i_j})$. The function $t : \Sigma^\star \longrightarrow \Sigma^\star$ translating proofs in h into proofs in Opt can be defined in the following way:

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

The word $Comp - G$ in the definition of t is the computation of G producing the code of D_{i_j}, $Comp - TEST_{K,x}$ is a computation of D_{i_j} accepting $TEST_{K,x}$.

In order to prove that t is polynomial time computable let us notice that $Comp - G$ is fixed and connected with K, so the only task is to prove that $Comp - TEST_{K,x}$ can be constructed in polynomial time. This follows from the local uniformity property of $TEST_{K,x}$ formulas and from the fact that D_{i_j} is deterministic and works in polynomial time.

5 Independence result

Let T be any formal theory whose language contains the language of arithmetic, i. e. the language $\{0, 1, \leq, =, +, \cdot\}$. 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.

The notation $T \vdash \beta$ means that a first order formula β is provable in T.

Let M be a deterministic Turing machine.

Definition 7. *We say that M is consistent if M accepts only satisfiable boolean formulas (if M accepts w, then $w \in SAT$).*

By "$L(M) \subset SAT$" we denote the first order formula which expresses the consistency of M , i.e. $\forall_{w \in L(M)}[w$ is a satisfiable boolean formula$]$.

Definition 8. *A deterministic Turing machine M is $T-consistent$ if and only if $T \vdash$ "$L(M) \subset SAT$".*

Definition 9. *A set $A \subset SAT$ is T-provably easy if there exists deterministic polynomial time clocked Turing machine M fulfilling (1) – (2)*

(1) M is $T - consistent$;
(2) L(M)=A.

Theorem 2 can be exploited to obtain the following independence result.

Theorem 3. *If there does not exist a p-optimal proof system for SAT, then for every theory T there exists an easy subset of SAT which is not T-provably easy.*

Proof. Suppose, on the contrary, that there exists a theory T such that all easy subsets of SAT are T-provably easy. Then the following recursively enumerable set of machines Ω_T=$\{M$: M is a deterministic polynomial time clocked Turing machine which is $T - consistent\}$ creates a recursive **P**-presentation of the class of all easy subsets of SAT. By Theorem 2, this implies that there exists a p-optimal proof system for SAT, giving a contradiction.

References

1. Balcazar, J.L., Díaz, J., Gabarró, J.: Structural complexity I. 2nd edn. Springer-Verlag, Berlin Heidelberg New York (1995)
2. Cook, S.A., Reckhow R.A.: The relative efficiency of propositional proof systems. J. Symbolic Logic 44 (1979) 36–50
3. Fenner, S., Fortnow, L., Naik, A., Rogers, R.: On inverting onto functions. In: Proc. 11th Annual IEEE Conference on Computational Complexity, (1996) 213 – 222
4. Hartmanis, J., Hemachandra, L.: Complexity classes without machines: On complete languages for UP. Theoret. Comput. Sci. 58 (1988) 129 – 142
5. Kowalczyk, W.: 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-Verlag, Berlin Heidelberg New York (1988) 364 – 369
6. Köbler, J., Messner, J.: Complete problems for promise classes by optimal proof systems for test sets. In: Proc. 13th Annual IEEE Conference on Computational Complexity, (1998) 132–140
7. Köbler, J., Messner, J.: Is the standard proof system for SAT p-optimal? In: Proc 20th Annual Conference on the Foundations of Software Technology and Theoretical Computer Science. Lecture Notes in Computer Science, Vol. 1974. Springer-Verlag, Berlin Heidelberg New York (2000) 361 – 372

8. Krajíček, J., Pudlák, P.: Propositional proof systems, the consistency of first order theories and the complexity of computations. J. Symbolic Logic 54 (1989) 1063–1079
9. Messner,J.: On optimal algorithms and optimal proof systems. In: Proc. 16th Symposium on Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, Vol. 1563. Springer-Verlag, Berlin Heidelberg New York (1999) 541 –550
10. Messner, J., Torán, J.: Optimal proof systems for Propositional Logic and complete sets. In: Proc. 15th Symposium on Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, Vol. 1373. Springer-Verlag, Berlin Heidelberg New York (1998) 477 – 487
11. Sadowski, Z.: On an optimal deterministic algorithm for SAT. In: Proc 12th International Workshop, CSL'98. Lecture Notes in Computer Science, Vol. 1584. Springer-Verlag, Berlin Heidelberg New York (1999) 179 – 187
12. Sipser, M.: On relativization and the existence of complete sets. In: Proc. Internat. Coll. on Automata, Languages and Programming. Lecture Notes in Computer Science, Vol. 140. Springer-Verlag, Berlin Heidelberg New York (1982) 523 –531