Characterizing the Existence of Optimal Proof Systems and Complete Sets for Promise Classes

Olaf Beyersdorff[1] and Zenon Sadowski[2]

[1] Institute of Theoretical Computer Science, Leibniz University Hannover, Germany
beyersdorff@thi.uni-hannover.de
[2] Institute of Mathematics, University of Białystok, Poland
sadowski@math.uwb.edu.pl

Abstract. In this paper we investigate the following two questions:
Q1: Do there exist optimal proof systems for a given language L?
Q2: Do there exist complete problems for a given promise class C?
For concrete languages L (such as TAUT or SAT) and concrete promise classes C (such as $\mathsf{NP}\cap\mathsf{coNP}$, UP, BPP, disjoint NP-pairs etc.), these questions have been intensively studied during the last years, and a number of characterizations have been obtained. Here we provide new characterizations for Q1 and Q2 that apply to almost all promise classes C and languages L, thus creating a unifying framework for the study of these practically relevant questions.
While questions Q1 and Q2 are left open by our results, we show that they receive affirmative answers when a small amount on advice is available in the underlying machine model. This continues a recent line of research on proof systems with advice started by Cook and Krajíček [6].

1 Introduction

A general proof system in the sense of Cook and Reckhow [7] can be understood as a nondeterministic guess-and-verify algorithm. The question whether there exist optimal or p-optimal proof systems essentially asks whether there exists the best such verification procedure. For practical purposes, such an optimal proof system would be extremely useful, as both the search for good verification algorithms as well as the quest for lower bounds to the proof size could concentrate on the optimal system. Thus the following question is of great significance:

Q1: Do there exist (p-)optimal proof systems for a given language L?

Posed by Krajíček and Pudlák [15], this question has remained unresolved for almost twenty years. Sufficient conditions were established by Krajíček and Pudlák [15] by $\mathsf{NE} = \mathsf{coNE}$ for the existence of optimal and $\mathsf{E} = \mathsf{NE}$ for p-optimal propositional proof systems, and these conditions were subsequently weakened by Köbler, Messner, and Torán [13]. Necessary conditions for a positive answer to Q1 are tightly linked to the following analogue of Q1 for promise complexity classes lacking an easy syntactic machine model:

Q2: Do there exist complete problems for a given promise class C?

Like the first question also Q2 has a long research record, dating back to the 80's when Kowalczyk [14] and Hartmanis and Hemachandra [12] considered this question for $\mathsf{NP} \cap \mathsf{coNP}$ and UP. This research agenda continues to recent days where, due to cryptographic and proof-theoretic applications, disjoint NP-pairs have been intensively studied (cf. [8, 9, 11, 2] and [10] for a survey).

As many computational tasks are formulated as function problems [20], it is also interesting to extend Q2 to function classes. In this formulation Q1 becomes a special case of Q2 because all proof systems for a given language can be understood as a promise function class in which complete functions correspond to p-optimal proof systems. In fact, Köbler, Messner, and Torán [13] have shown that, with respect to Q2, proof systems provide the most difficult instances among all promise classes, i.e., a positive answer to Q1 implies a positive answer for Q2 for many choices of L and C.

In the present paper we continue this line of research. While Köbler, Messner, and Torán [13] focused on the implication Q1 $\Rightarrow$ Q2, we provide new characterizations for both Q1 and Q2. In fact, from these characterizations we can also easily read off the implication Q1 $\Rightarrow$ Q2 (under suitable assumptions), thus in addition, we provide alternative proofs for some results of [13]. Köbler, Messner, and Torán used the notion of a test set to measure the complexity of the promise. Here we pursue a different but related approach by representing the promise in a language L and then using a proof system for L to verify the promise. On the propositional level, such representations have been successfully used to express the consistency of propositional proof systems (known as the reflection principle, cf. [5, 15]) or the disjointness of NP-pairs [16, 2]. We create a unifying framework which generalizes these methods to arbitrary languages.

We will now describe in more detail our results and the organization of the paper. After developing the notion of representations in Sects. 2 and 3 we examine Q1 in Sect. 4 where we prove that a language L has a p-optimal proof system if and only if all polynomial-time computable subsets of L are recursively enumerable. A similar characterization also holds for the existence of optimal proof systems. This widely generalizes previous results from [18] for propositional proof systems and provides interesting characterizations for a number of applications like the graph isomorphism and automorphism problems.

In Sect. 5 we proceed with question Q2 where we discuss a characterization of Q2 in terms of uniform enumerations of promise obeying machines. Section 6 then contains our results on the connections between Q1 and Q2. We show that, under suitable assumptions, a promise class C has complete problems if and only if there exists a proof system for some language L in which C is representable. This also yields a general method to show the equivalence of reductions of varying strength with respect to Q2. In addition, we obtain that L has a p-optimal proof system if and only if every promise class expressible in L has a complete set or function. Different versions of these results hold for both optimality and p-optimality. We also apply these general theorems to concrete promise classes like UP, $\mathsf{NP} \cap \mathsf{coNP}$, and disjoint NP-pairs.

Finally, in Sect. 7 we show that the relation between proof systems and promise classes also holds in the presence of advice. Employing recent advances of Cook and Krajíček [6] who show that optimal propositional proof systems exist which use only one bit of advice, we obtain complete sets for a large number of promise classes when advice is available.

Due to space restrictions we sketch or omit proofs in this extended abstract.

2 Preliminaries

We assume basic familiarity with complexity classes (cf. [1]). Our basic model of computation are polynomial-time Turing machines and transducers. Tacitly we assume these machines to be suitably encoded by strings. We also assume that they always have a polynomial-time clock attached bounding their running time such that this running time is easy to detect from the code of the machine.

For a language L and a complexity class C, the set of all C-*easy subsets* of L consists of all sets $A \subseteq L$ with $A \in \mathsf{C}$. A class C of languages has a *recursive* P-*presentation* (resp. NP-presentation) if there exists a recursively enumerable list $N_1, N_2, \ldots$ of (non-)deterministic polynomial-time clocked Turing machines such that $L(N_i) \in \mathsf{C}$ for $i \in \mathbb{N}$, and, conversely, for each $A \in \mathsf{C}$ there exists an index i with $A \subseteq L(N_i)$. In this definition, it would also be natural to replace $A \subseteq L(N_i)$ by the stronger requirement $A = L(N_i)$, but the weaker concept suffices for our purpose.

Proof Systems. Cook and Reckhow [7] defined the notion of a *proof system* for a language L quite generally as a polynomial-time computable function f with range L. A string w with $f(w) = x$ is called an f-proof for $x \in L$. By $f \vdash_{\leq m} x$ we indicate that x has an f-proof of size $\leq m$. For a subset $A \subseteq L$ we write $f \vdash_* A$ if there is a polynomial p such that $f \vdash_{\leq p(|x|)} x$ for all $x \in A$.

Proof systems are compared by simulations [7, 15]. If f and g are proof systems for L, we say that g *simulates* f (denoted $f \leq g$), if there exists a polynomial p such that for all $x \in L$ and f-proofs w of x there is a g-proof w' of x with $|w'| \leq p(|w|)$. If such a proof w' can even be computed from w in polynomial time, we say that g *p-simulates* f (denoted $f \leq_p g$). A proof system for L is called *(p-)optimal* if it (p-)simulates all proof systems for L.

Promise Classes. Following the approach of Köbler, Messner, and Torán [13], we define promise classes in a very general way. A promise R is described as a binary predicate between nondeterministic polynomial-time Turing machines N and strings x, i.e., $R(N, x)$ means that N obeys promise R on input x. A machine N is called an R-*machine* if N obeys R on any input $x \in \Sigma^*$. Given a promise predicate R, we define the language class $\mathsf{C}_R = \{L(N) \mid N \text{ is an R-machine}\}$ and call it the promise class generated by R. Instead of R-machines we will also speak of C_R-machines. Similarly, we define function promise classes by replacing $L(N)$ by the function computed by N (cf. [13]). For functions we use the following

variant of many-one reductions (cf. [13]): $f \leq g$ if there exists a polynomial-time computable function t such that $f(x) = g(t(x))$ for all x in the domain of f.

In this general framework it is natural to impose further restrictions on promise classes. One assumption which we will make throughout the paper is the presence of *universal machines*, i.e., we only consider promise conditions R such that there exists a universal machine U_R which, given an R-machine N, input x, and time bound 0^m, efficiently simulates $N(x)$ for m steps such that U_R obeys promise R on $\langle N, x, 0^m \rangle$.

Occasionally, we will need that C-machines can perform nondeterministic polynomial-time computations without violating the promise. We make this precise via the following notion from [13]: for a complexity class A and a promise class C defined via promise R, we say that A-*assertions are useful for* C if for any language $A \in \mathsf{A}$ and any nondeterministic polynomial-time Turing machine N the following holds: if N obeys promise R on any $x \in A$, then there exists a language $C \in \mathsf{C}$ such that $C \cap A = L(N) \cap A$. A similar definition also applies for function classes. Throughout this paper we will only consider promise classes C for which P-assertions are useful. If also NP-assertions are useful for C, then we say that C *can use nondeterminism.*

The set of all proof systems for a language L is an example for a promise function class, where the promise for a given function f is $rng(f) = L$. We define a larger class $PS(L)$ where we only concentrate on correctness but not on completeness of proof systems. This is made precise in the following definition.

Definition 1. *For a language L, the promise function class $PS(L)$ contains all polynomial-time computable functions f with $rng(f) \subseteq L$.*

3 Representations

In order to verify a promise, we need appropriate encodings of promise conditions. In the next definition we explain how a promise condition for a machine can be expressed in an arbitrary language.

Definition 2. *A promise R is* expressible in a language L *if there exists a polynomial-time computable function $corr : \Sigma^* \times \Sigma^* \times 0^* \to \Sigma^*$ such that the following conditions hold:*

1. Correctness: *For every Turing machine N, for every $x \in \Sigma^*$ and $m \in \mathbb{N}$, if $corr(x, N, 0^m) \in L$, then N obeys promise R on input x.*
2. Completeness: *For every R-machine N with polynomial time bound p, the set $Correct(N) = \{ corr(x, N, 0^{p(|x|)}) \mid x \in \Sigma^* \}$ is a subset of L.*
3. Local recognizability: *For every Turing machine N, the set $Correct(N)$ is polynomial-time decidable.*

We say that the promise class C generated by R is expressible in L if R is expressible in L. If the elements $corr(x, N, 0^m)$ only depend on $|x|$, N, and m, but not on x, we say that C is expressible in L by a length-depending *promise.*

This definition applies to both language and function promise classes. One of the most important applications for the above concept of expressibility is to use $L = \mathrm{TAUT}$. Expressing promise conditions by propositional tautologies is a well known approach with a long history. For propositional proof systems, leading to the promise function class $PS(\mathrm{TAUT})$, propositional expressions are constructed via the reflection principle of the proof system (cf. [5, 15]). Propositional expressions have also been used for other promise classes like disjoint NP-pairs and its generalizations [2, 3]. Typically, these expressions are even length depending. We remark that Köbler, Messner, and Torán [13] have used a related approach, namely the notion of a test set, to measure the complexity of promise conditions.

As a first example, consider the set of all P-easy subsets of a language L. The next lemma shows that this promise class is always expressible in L.

Lemma 3. *For every language L, the P-easy subsets of L are expressible in L.*

Using expressibility of a promise class in a language L, we can verify the promise for a given machine with the help of short proofs in some proof system for L. This leads to the following concept:

Definition 4. *Let C be a promise class which is expressible in a language L. Let further A be a language from C and P be a proof system for L. We say that A is* representable in P *if there exists a C-machine N for A such that $P \vdash_* Correct(N)$. If these P-proofs of $corr(x, N, 0^{p(|x|)})$ can even be constructed from input x in polynomial time, then we say that A is* p-representable in P.

Furthermore, if every language $A \in \mathsf{C}$ is (p-)representable in P, then we say that C is (p-)representable in P.

Intuitively, representability of A in P means that we have short P-proofs of the promise condition of A (with respect to some C-machine for A). Given a proof system P for L and a promise class C which is expressible in L, it makes sense to consider the subclass of all languages or functions from C which are representable in P. This leads to the following definition:

Definition 5. *For a promise class C expressible in a language L and a proof system P for L, let $\mathsf{C}(P)$ denote the class of all $A \in \mathsf{C}$ which are representable in P.*

Note that for each $A \in \mathsf{C}$ there exists some proof system P for L such that $A \in \mathsf{C}(P)$, but in general $\mathsf{C}(P)$ will be a strict subclass of C which enlarges for stronger proof systems. It is, of course, interesting to ask whether these subclasses $\mathsf{C}(P)$ have sufficiently good properties. In particular, it is desirable that $\mathsf{C}(P)$ is closed under reductions. Therefore, we make the following definition:

Definition 6. *A promise class C is* provably closed under a reduction $\leq_R$ in L *if C is expressible in L and for each proof system P for L there exists a proof system P' for L such that $P \leq P'$ and for all $A \in \mathsf{C}$ and $B \in \mathsf{C}(P')$, $A \leq_R B$ implies $A \in \mathsf{C}(P')$.*

We remark that provable closure of C under $\leq_R$ is a rather weak notion as it does not even imply closure of C under $\leq_R$ in the ordinary sense (because of the

restriction $A \in \mathsf{C}$). Also we do not require each subclass $\mathsf{C}(P)$ to be closed under $\leq_R$, but that for each proof system P this holds for some stronger system P'. This is a sensible requirement, because proof systems for L can be defined quite arbitrarily, and closure of $\mathsf{C}(P)$ typically requires additional assumptions on P (cf. [2] where provable closure of the class of disjoint NP-pairs under different reductions is shown). In fact, it is not difficult to construct counterexamples:

Proposition 7. *Let C be a promise class which is expressible in a language L and let $\leq_R$ be a reduction for C. Let further P be a proof system for L such that there exist $A, B \in \mathsf{C} \setminus \mathsf{C}(P)$ with $A \leq_R B$. Then there exists a proof system $P' \geq P$ such that $\mathsf{C}(P')$ is not closed under $\leq_R$.*

4 Optimal Proof Systems and Easy Subsets

In this section we search for characterizations for the existence of optimal or even p-optimal proof systems for arbitrary languages L (Question Q1) and apply these results to concrete choices for L. We start with a criterion for the existence of p-optimal proof systems.

Theorem 8. *Let L be a language such that $PS(L)$ is expressible in L. Then L has a p-optimal proof system if and only if the P-easy subsets of L have a recursive P-presentation.*

Proof (Idea). For the forward direction, we observe that every P-easy subset of L has short proofs in some proof system for L. These proofs are translated into short proofs in the p-optimal proof system by some polynomial-time Turing transducer. Thus, by enumerating all polynomial-time clocked Turing transducers, we can construct a recursive P-presentation of all P-easy subsets of L.

Conversely, we construct a p-optimal proof system P_{opt} in the following way. A P_{opt}-proof of a is of the form $\langle \pi, P, \mathit{certificate} \rangle$, where P is a polynomial-time clocked transducer such that $P(\pi) = a$. The certificate assures that $P(\pi) \in L$. It follows from expressibility of $PS(L)$ in L that $\mathit{Correct}(P)$ is a P-easy subset of L if and only if P produces only elements from L (for any input). Hence, we can use P-presentability of the P-easy subsets of L to produce certificates. □

By a similar argument we can provide two characterizations for the existence of optimal proof systems.

Theorem 9. *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists an optimal proof system for L.*
2. *The NP-easy subsets of L have a recursive NP-presentation.*
3. *The P-easy subsets of L have a recursive NP-presentation.*

Given these general results, it is interesting to ask for which languages L the set $PS(L)$ is expressible in L. Our next lemma provides sufficient conditions:

Lemma 10. *Let L be a language fulfilling the following two conditions:*

1. *Natural numbers can be encoded by elements of L, i.e., there exists an injective function $Num : \mathbb{N} \to L$ which is both computable and invertible in polynomial time.*
2. *L possesses an AND-function, i.e., there exists a function $AND : \Sigma^* \times \Sigma^* \to \Sigma^*$ which is both polynomial-time computable and polynomial-time invertible such that for all $x, y \in \Sigma^*$, $AND(x, y) \in L$ if and only if $x \in L$ and $y \in L$.*

Then $PS(L)$ is expressible in L.

Using this lemma we can show L-expressibility of $PS(L)$ for many interesting choices of L:

Proposition 11. *For any of the following languages L, the set $PS(L)$ is expressible in L:*

- SAT_i *for $i \in \mathbb{N}$ (the satisfiability problem for quantified propositional formulas with i quantifier alternations, starting with existential quantifiers),*
- TAUT_i *for $i \in \mathbb{N}$ (quantified propositional tautologies with i quantifier alternations, starting with universal quantifiers),*
- QBF *(quantified propositional tautologies),*
- *the graph isomorphism problem* GI*, its complement* $\overline{\mathrm{GI}}$*, and the complement* $\overline{\mathrm{GA}}$ *of the graph automorphism problem.*

For GI, which like any problem in NP has an optimal proof system, we obtain the following characterization on the existence of a p-optimal proof system.

Corollary 12. GI *has a p-optimal proof system if and only if there exists a recursive* P*-presentation of all polynomial-time computable subsets of* GI.

Let us remark that in Lemma 10, instead of an AND-function we could also use a padding function for L. In this way we obtain a similar result as Corollary 12 for GA (which is not known to possess an AND-function).

5 Complete Sets and Enumerations

In this section we consider the question Q2, asking whether language or function promise classes have complete sets or functions. There is a long history of equating complete sets and recursive enumerations of machines. The following result essentially stems from [13], but particular cases of the theorem have been been previously obtained, namely for $\mathsf{NP} \cap \mathsf{coNP}$ by Kowalczyk [14], for UP by Hartmanis and Hemachandra [12], and, more recently, for disjoint NP-pairs by Glaßer, Selman, and Sengupta [8]. We just formulate the theorem for language classes, but a similar result also holds for promise function classes.

Theorem 13 (Köbler, Messner, Torán [13]). *Let C be a promise class which is closed under many-one reductions. Then C has a many-one complete problem if and only if there exists a recursive enumeration $(N_i)_{i \geq 0}$ of C-machines such that $\mathsf{C} = \{L(N_i) \mid i \geq 0\}$.*

Let us note that in the proof of the forward implication of Theorem 13, the hypothesis that C is closed under many-one reductions seems indeed crucial. Namely, if C consists of all P-easy subsets of TAUT, then C trivially contains a many-one complete set. On the other hand, a recursive enumeration of C-machines as in Theorem 13 is rather unlikely to exist, as this would imply the existence of a p-optimal propositional proof system by Theorem 8. But of course, the P-easy subsets of TAUT are not closed under many-one reductions.

6 Optimal Proof Systems and Complete Sets

Now we are ready to analyse the relations between our central questions Q1 and Q2 on the existence of optimal proof systems for languages L and the existence of complete sets for promise classes C. While Köbler, Messner, and Torán [13] have shown that for many natural choices of L and C, a positive answer to Q1 implies a positive answer to Q2, we will provide here a number of characterizations involving both questions. In particular, these characterizations will also yield the above mentioned relation between Q1 and Q2 for concrete applications.

Our first result characterizes the existence of complete sets for a promise class C by the representability of C in a proof system.

Theorem 14. *Let C be a promise language (or function) class which can use nondeterminism and let L be a language such that C is provably closed under many-one reductions in L. Then C has a many-one complete language (or function) if and only if there exists a proof system for L in which C is representable.*

Proof (Idea). For the forward implication, we code a many-one complete language A for C into some proof system P for L. By provable closure under reductions, L has a proof system $P' \geq P$ such that $\mathsf{C}(P')$ is closed under many-one reductions. As $A \in \mathsf{C}(P')$ and A is many-one complete for C, we get $\mathsf{C}(P') = \mathsf{C}$.

Conversely, let P be a proof system for L in which C is representable. Using the universal machine for C, we construct a complete set for C by simulating C-machines N on their inputs. But before we start such a simulation, we check the promise of N by guessing short P-proofs for $\mathit{Correct}(N)$. For this last step we need that C can use nondeterminism. □

For promise classes not using nondeterminism we obtain the following result:

Theorem 15. *Let C be a promise language (or function) class which is closed under many-one reductions and let L be a language such that C is expressible in L. Then C has a many-one complete language (or function) if and only if L has a proof system in which C is p-representable.*

Let us mention some applications of this result. The promise class DisjNP of disjoint NP-pairs and the class UP are expressible in TAUT, and the class $\mathsf{NP} \cap \mathsf{coNP}$ is expressible in QBF (cf. [2, 13, 17, 19]). Hence we obtain the following corollary exemplifying our theorem.

Corollary 16.

1. *Complete disjoint* NP*-pairs exist if and only if* TAUT *has a proof system in which* DisjNP *is p-representable (if and only if* TAUT *has a proof system in which* DisjNP *is representable).*
2. UP *has a complete language if and only if* TAUT *has a proof system in which* UP *is p-representable.*
3. NP ∩ coNP *has a complete language if and only if* QBF *has a proof system in which* NP ∩ coNP *is p-representable.*

Theorem 14 also allows to derive results which show that the question of the existence of complete problems for C does not depend on the strength of the underlying reduction. This can be done as in the following corollary:

Corollary 17. *Let* $\leq$ *and* $\leq'$ *be two reductions which are refined by many-one reductions. Assume further that* C *can use nondeterminism and is both provably closed under* $\leq$ *and* $\leq'$ *in some language* L*. Then* C *has a* $\leq$*-complete problem if and only if* C *has a* $\leq'$*-complete problem.*

In this way it can be shown, for example, that the question of the existence of complete disjoint NP-pairs is equivalent for reductions ranging from strong many-one reductions to smart Turing reductions (cf. [8, 2]).

Our next result shows that question Q1 on the existence of p-optimal proof systems for a language L can be characterized by a "universally quantified" version of the condition from Theorem 15. Further, Q1 is even equivalent to the existence of complete sets for all promise classes representable in L:

Theorem 18. *Let* L *be a language such that* $PS(L)$ *is expressible in* L*. Then the following conditions are equivalent:*

1. *There exists a p-optimal proof system for* L*.*
2. *There exists a proof system for* L *in which any promise class which is expressible in* L *is p-representable.*
3. *There exists a proof system for* L *in which the class of all* P*-easy subsets of* L *is p-representable.*
4. *Every promise language and function class which is expressible in* L *has a many-one complete language or function.*

Proof (Sketch). The proof is structured into the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$ and $2 \Rightarrow 4 \Rightarrow 1$. For $1 \Rightarrow 2$, let P be a p-optimal proof system for L and let C be a promise class expressible in L. For each $A \in \mathsf{C}$ and each C-machine N for A we can construct a proof system P' with short P'-proofs of $Correct(N)$. Translating these proofs into the p-optimal system P, we obtain $A \in \mathsf{C}(P)$.

Implication $2 \Rightarrow 3$ follows from Lemma 3. For the direction $3 \Rightarrow 1$, we need to construct from item 3 a recursive P-presentation of all P-easy subsets of L as in Theorem 8. This in turn yields a p-optimal proof system for L.

The equivalence between items 2 and 4 is the mentioned "universally quantified" version of Theorem 15. Finally, for $4 \Rightarrow 1$ we use the assumption of

expressibility of $PS(L)$ in L. As $PS(L)$ is a promise function class, item 4 guarantees the existence of a many-one complete function for $PS(L)$, which coincides with the notion of a p-optimal proof system for L. □

The next theorem contains a similar statement for optimal proof systems.

Theorem 19. *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists an optimal proof system for L.*
2. *L has a proof system P such that every promise class which is expressible in L is representable in the system P.*
3. *L has a proof system in which all P-easy subsets of L are representable.*

Combining Theorems 14 and 19 we obtain the following corollary which is essentially contained in [13].

Corollary 20. *Let L be a language. If L has an optimal proof system, then any promise language or function class C which is expressible in L and which can use nondeterminism has a complete language or function.*

As the proof of the backward implication of Theorem 14 does not use provable closure of C under reductions in L, we can formulate Corollary 20 without this assumption.

Comparing Theorem 18 and Corollary 20, it is apparent that while we could prove the equivalence of the existence of p-optimal proof systems for L and complete problems for all promise classes expressible in L (Theorem 18), we did not obtain this equivalence for optimal proof systems (cf. Corollary 20). The reason is that $PS(L)$, considered as a promise function class, does not seem to have the property to use nondeterminism, because otherwise, the existence of an optimal proof system for L would already imply the existence of a p-optimal proof system for L. We can even obtain a slightly stronger result:

Proposition 21. *If $PS(\mathrm{SAT})$ can use nondeterminism, then every language with an optimal proof system also has a p-optimal proof system.*

Proof. Assume that $PS(\mathrm{SAT})$ can use nondeterminism. By Proposition 11, the class $PS(\mathrm{SAT})$ is expressible in SAT. As SAT has an optimal proof system, Corollary 20 now yields a complete function for $PS(\mathrm{SAT})$ which coincides with the notion of a p-optimal proof system for SAT. From this we conclude that every language with an optimal proof system also has a p-optimal proof system by a result from [3]. □

7 Optimal Proof Systems with Advice

Whether or not there exist optimal proof systems or complete sets for promise classes remains unanswered by our results above. Hence, our central questions Q1 and Q2 remain open. As these problems have been open for more than twenty

years by now, many researchers tend to believe in a negative answer (of course, this is arguable, but in the algorithmic world negative results are usually harder to obtain than positive ones).

Recently, Cook and Krajíček [6] have introduced the concept of propositional proof systems with advice which seems to yield a strictly more powerful model than the classical Cook-Reckhow setting. Surprisingly, Cook and Krajíček [6] have shown that in the presence of advice, optimal propositional proof systems exist (cf. also [4] for a generalization to arbitrary languages). Our next result shows that the relation between optimal proof systems and complete sets for promise classes can be transferred to the advice setting. Thus we derive from Cook and Krajíček's results the following strong information on complete problems in the presence of advice.

Theorem 22. *Let* C *be a promise complexity class and let* L *be a language such that* C *is expressible in* L *by a length-depending promise. Then* $\mathsf{C}/1$ *contains a problem (or function) using one bit of advice which is many-one hard for* C*.*

Proof (Sketch). Let $\langle \cdot, \ldots, \cdot \rangle$ be a polynomial-time computable length-injective tupling function. We now define the problem (or function) A_{C} with one advice bit which will be many-one hard for C. Inputs are of the form $\langle x, 0^N, 0^m \rangle$ where x is the input, 0^N is the unary encoding of a Turing machine N, and 0^m is the time bound for N. At such an input, A_{C} first computes the string $corr(x, N, 0^m)$. Then A_{C} uses its advice bit to verify whether or not $corr(x, N, 0^m)$ is in L (for this step we could have also used the optimal proof system for L with one bit of advice, cf. [6, 4]). If $corr(x, N, 0^m) \in L$, then A_{C} simulates N on input x for at most m steps and produces the corresponding output (in case the simulation does not terminate it rejects or outputs some fixed element). As $\langle \cdot, \ldots, \cdot \rangle$ is length injective and $corr$ is length depending, the element $corr(x, N, 0^m)$ is uniquely determined by $|\langle x, 0^N, 0^m \rangle|$ and therefore the advice bit of A_{C} can in fact refer to $corr(x, N, 0^m)$.

If A is a problem (or function) from C and N is a C-machine for A with polynomial running time p, then A many-one reduces to A_{C} via $x \mapsto \langle x, 0^N, 0^{p(|x|)} \rangle$. Hence A_{C} is many-one hard for C. □

Let us state a concrete application of this general result. As disjoint NP-pairs are expressible in TAUT by a length-depending promise [2], we obtain:

Corollary 23. *There exist a disjoint pair* (A, B) *and a sequence* $(a_n)_{n \in \mathbb{N}}$ *with the following properties:*

1. *A and B are computable in nondeterministic polynomial time with advice a_n for inputs of length n.*
2. *The set* $\{ \langle a_n, 0^n \rangle \mid n \in \mathbb{N} \}$ *is computable in* coNP*.*
3. *Every disjoint* NP*-pair is polynomial-time many-one reducible to* (A, B)*.*

Acknowledgements. We thank the anonymous referees for helpful comments and detailed suggestions on how to improve this paper.

References

1. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin Heidelberg, 1988.
2. O. Beyersdorff. Classes of representable disjoint NP-pairs. *Theoretical Computer Science*, 377(1–3):93–109, 2007.
3. O. Beyersdorff, J. Köbler, and J. Messner. Nondeterministic functions and the existence of optimal proof systems. Submitted to *Theoretical Computer Science*.
4. O. Beyersdorff, J. Köbler, and S. Müller. Nondeterministic instance complexity and proof systems with advice. In *Proc. 3rd International Conference on Language and Automata Theory and Applications*, volume 5457 of *Lecture Notes in Computer Science*, pages 164 – 175. Springer-Verlag, Berlin Heidelberg, 2009.
5. S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proc. 7th Annual ACM Symposium on Theory of Computing*, pages 83–97, 1975.
6. S. A. Cook and J. Krajíček. Consequences of the provability of NP $\subseteq$ P/poly. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.
7. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
8. C. Glaßer, A. L. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. *Information and Computation*, 200(2):247–267, 2005.
9. C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.
10. C. Glaßer, A. L. Selman, and L. Zhang. Survey of disjoint NP-pairs and relations to propositional proof systems. In O. Goldreich, A. L. Rosenberg, and A. L. Selman, editors, *Essays in Theoretical Computer Science in Memory of Shimon Even*, pages 241–253. Springer-Verlag, Berlin Heidelberg, 2006.
11. C. Glaßer, A. L. Selman, and L. Zhang. Canonical disjoint NP-pairs of propositional proof systems. *Theoretical Computer Science*, 370(1–3):60–73, 2007.
12. J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.
13. J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.
14. W. Kowalczyk. Some connections between representability of complexity classes and the power of formal systems of reasoning. In *Proc. 11th Symposium on Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 364–369. Springer-Verlag, Berlin Heidelberg, 1984.
15. J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.
16. J. Krajíček and P. Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.
17. Z. Sadowski. On an optimal quantified propositional proof system and a complete language for NP $\cap$ co-NP. In *Proc. 11th International Symposium on Fundamentals of Computing Theory*, volume 1279 of *Lecture Notes in Computer Science*, pages 423–428. Springer-Verlag, Berlin Heidelberg, 1997.
18. Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets of TAUT. *Theoretical Computer Science*, 288(1):181–193, 2002.
19. Z. Sadowski. Optimal proof systems and complete languages. In *Local Proc. 4th Conference on Computability in Europe*, pages 407–414. University of Athens, 2008.
20. A. L. Selman. Much ado about functions. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, pages 198–212, 1996.