Electronic Colloquium on Computational Complexity, Report No. 81 (2009)

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 Hanover, 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 of 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 [7].

1 Introduction

A general proof system in the sense of Cook and Reckhow [8] 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 [17], this question has remained unresolved for almost twenty years. Sufficient conditions were established by Krajíček and Pudlák [17] 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 [14]. 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?

* An extended abstract of this paper appeared in the proceedings of the conference CSR 2009 [5].

ISSN 1433-8092

Like the first question also Q2 has a long research record, dating back to the 80's when Kowalczyk [16] and Hartmanis and Hemachandra [13] 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. [9, 10, 12, 2] and [11] for a survey).

As many computational tasks are formulated as function problems [22], 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 [14] 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 [14] 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 [14]. 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. [6, 17]) or the disjointness of NP-pairs [18, 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 [20] 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 [7] 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.

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 [8] 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 [8, 17]. 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 [14], 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. [14]). For functions we use the following variant of many-one reductions (cf. [14]): $f \leq g$ 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 [14]: 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. Namely, A-assertions are useful for a function class C if for any language $A \in \mathsf{A}$ and any polynomial-time clocked Turing transducer N it holds: if N obeys promise R on any input $x \in A$, then there exists a function $f \in C$ such that $N(x) = f(x)$ for any $x \in A$. 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 choose L as the set of propositional tautologies 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 $PS(\text{TAUT})$, propositional expressions are constructed via the reflection principle of the proof system (cf. [6, 17]). 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 [14] 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.*

Proof. Let N be a deterministic polynomial-time Turing machine with running time p. We define the function $corr(N, x, 0^m)$ as

$$corr(x, N, 0^m) = \begin{cases} x & \text{if } N(x) \text{ accepts in} \leq m \text{ steps} \\ x_0 & \text{otherwise} \end{cases}$$

with some fixed element $x_0 \notin L$. □

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* $\leq_R$.

Proof. Under the hypotheses of the proposition, we construct the proof system P' as follows. We choose a C-machine N for B and define

$$P'(y) = \begin{cases} x & \text{if } y = 0x \text{ and } x \in \mathit{Correct}(N) \\ P(x) & \text{if } y = 1x \\ x_0 & \text{otherwise} \end{cases}$$

where x_0 is a fixed element from L. As $P' \vdash_* \mathit{Correct}(N)$ we have $\mathsf{C}(P') = \mathsf{C}(P) \cup \{B\}$. In particular, A is not contained in $\mathsf{C}(P')$, but $B \in \mathsf{C}(P')$ and $A \leq_R B$. Therefore $\mathsf{C}(P')$ 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. Let f be a p-optimal proof system for L and let A be a polynomial-time computable subset of L. We can define a proof system f_A for L as follows:

$$f_A(x) = \begin{cases} f(y) & \text{if } x = 0y \\ a & \text{if } x = 1a \text{ and } a \in A \\ b & \text{otherwise} \end{cases}$$

where b is a fixed element in L. Because f is p-optimal, f_A is p-simulated by f via some polynomial-time computable function t_A.

As this can be done for all P-easy subsets A of L, we get a recursive P-presentation of L as follows. Let $(t_i)_{i\in\mathbb{N}}$ be an enumeration of all deterministic polynomial-time clocked Turing transducers. For $i \in \mathbb{N}$ consider the following set of algorithms M_i:

```
1  Input: x
2  IF f(t_i(1x)) = x THEN accept ELSE reject
```

Apparently, these algorithms M_i can be computed by deterministic polynomial-time Turing machines. Further, each M_i only accepts inputs from L because if M_i accepts x, then we have an f-proof for x.

Now for each P-easy subset A of L, some machine computing the above function t_A appears in the enumeration t_i, and therefore A is accepted by M_i for the appropriate index i such that t_i computes t_A. Therefore M_i is a recursive P-presentation of the class of all P-easy subsets of L.

For the converse direction, let $(M_i)_{i\in\mathbb{N}}$ be a recursive P-presentation of the P-easy subsets of L. We construct a p-optimal proof system P_{opt} for L as follows. Inputs for P_{opt} are tuples

$$\langle \pi, P, 0^m, i, 0^n \rangle \ .$$

On such an input, P_{opt} first checks whether P is the encoding of a Turing transducer with a polynomial-time bound attached. If this is not the case, then P_{opt} outputs some fixed element $x_0 \in L$. Otherwise, P_{opt} spends n steps to compute the machine M_i from the enumeration $M_1, M_2 \ldots$ If n steps do not suffice to construct M_i, we output again $x_0 \in L$. Otherwise, P_{opt} computes $corr(\pi, P, 0^m)$ and checks whether M_i accepts $corr(\pi, P, 0^m)$ in n steps. Again, if M_i does not stop in $\leq n$ steps, then we output x_0. If P and π pass the test, then P_{opt} simulates P on input π and outputs $P(\pi)$.

Apparently, P_{opt} can be computed in polynomial time. For each Turing transducer N with running time p and each input x with $N(x) \in L$, the element $corr(x, N, 0^{p(|x|)})$ is contained in some polynomial-time computable subset of L. Therefore, P_{opt} is a proof system for L, because by the correctness and completeness conditions from Definition 2, the range of P_{opt} is exactly L.

To prove the p-optimality of P_{opt}, let P be a proof system for L. Because by assumption $PS(P)$ is expressible in L, the set $Correct(P)$ is a P-easy subset of L (by the local recognizability condition from Definition 2). Hence there exists an index i such that M_i decides $Correct(P)$. Let c be a constant such that M_i can be computed from i in time c and let p and q be polynomial time bounds for P and M_i, respectively. Then P is easily seen to be p-simulated by

$$\pi \mapsto \langle \pi, P, 0^{p(|\pi|)}, i, 0^{q(|corr(\pi, P, 0^{p(|\pi|)})|)+c} \rangle$$

which completes the proof. □

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 $PS(L)$ of all proof systems for 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.

Proof. We have to define the function *corr* according to Definition 2. Given a string x, an encoding of a polynomial-time computable Turing transducer N, and a number $m \in \mathbb{N}$, we first simulate $N(x)$ for $\leq m$ steps. Let y be the output of $N(x)$, if the simulation succeeded. Otherwise, we choose a fixed string $y \notin L$.

Next we interpret the binary encoding of N as a natural number (which we again denote by N) and compute $Num(N)$. We then define the function *corr* as

$$corr(x, N, 0^m) = AND(y, AND(Num(N), Num(m))) \ .$$

Clearly, *corr* is polynomial-time computable. To verify the conditions of Definition 2 for *corr*, we observe that correctness and completeness of *corr* follow because the string $w := AND(Num(N), Num(m))$ is contained in L for all $N, m \in \mathbb{N}$, and therefore $AND(y, w) \in L$ if and only if $y \in L$.

Local recognizability for *corr* follows as *AND* and *Num* are invertible in polynomial time and therefore for each polynomial-time Turing transducer N, the set $Correct(N)$ is in P. □

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.*

Proof. We have to check the conditions from the previous lemma. For languages consisting of formulas like SAT_i, TAUT_i, or QBF, the AND-function is provided by the Boolean connective $\wedge$. The function *Num* can be defined for example by

$$n \mapsto (p \vee \neg p) \wedge \cdots \wedge (p \vee \neg p) \qquad (n \text{ times}) \ ,$$

where p is a fixed propositional variable.

For GI, $Num(n)$ can be implemented by a pair (K_n, K_n) of cliques of size n. For $\overline{\mathrm{GI}}$ we take (K_n, K_{n+1}), and for $\overline{\mathrm{GA}}$ we take an easy rigid graph with n vertices. It is well known that GI has an AND-function (cf. [15]). For the AND-functions of $\overline{\mathrm{GI}}$ and $\overline{\mathrm{GA}}$ we can take the OR-functions of GI and GA (cf. [15]). □

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 [14], but particular cases of the theorem have been been previously obtained, namely for $\mathsf{NP} \cap \mathsf{coNP}$ by Kowalczyk [16], for UP by Hartmanis and Hemachandra [13], and, more recently, for disjoint NP-pairs by Glaßer, Selman, and Sengupta [9]. We just formulate the theorem for language classes, but a similar result also holds for promise function classes. The theorem is already included in [14], but for the benefit of the reader we include a full proof (our argument is more direct than the proof given in [14]).

Theorem 13 (Köbler, Messner, Torán [14]). *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* $\mathsf{C} = \{L(N_i) \mid i \geq 0\}$.

Proof. For the proof of the forward implication, let A_{C} be a many-one complete problem for C and let M be a C-machine for A_{C}. We fix an enumeration $N_1, N_2, \ldots$ of all polynomial-time Turing transducers. By assumption, the class C is closed under many-one reductions, and hence $M \circ N_i$ is a recursive enumeration of C.

For the converse implication, let f be a recursive function computing an enumeration $(N_i)_{i\in\mathbb{N}}$ of C-machines such that $\mathsf{C} = \{L(N_i) \mid i \geq 0\}$. We construct

a many-one complete problem A_{C} for C as follows: elements of A_{C} are of the form

$$\langle x, i, 0^m \rangle \ .$$

On such an input, A_{C} first simulates f on input i for at most m steps. If this simulation does not terminate, then we reject. Otherwise, $f(i)$ is guaranteed to output a C-machine N_i. Now we simulate this machine N_i on input x for at most m steps. Again, if the simulation does not stop, we reject. Otherwise, A_{C} answers according to the answer obtained in the simulation of $N_i(x)$.

By our general assumption on promise classes we have a universal C-machine, and therefore $A_{\mathsf{C}} \in \mathsf{C}$. To prove the hardness of A_{C} for C, let $A \in \mathsf{C}$. Then there exists a C-machine N_i in the enumeration computed by f such that N_i accepts A. Let p be a polynomial bounding the running time of N_i and let c be the time that $f(i)$ spends to compute N_i. Then the polynomial-time computable function

$$x \mapsto \langle x, i, 0^{p(|x|)+c} \rangle$$

many-one reduces A to A_{C}. □

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 [14] 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. For the proof of the forward implication, let C be a promise complexity class with a many-one complete language A. Let L be a language such that C is provably closed under reductions in L. Let P be an arbitrary proof system

for L and let N be a C-machine for A. We construct a proof system P' with $A \in \mathsf{C}(P')$ as follows:

$$P'(y) = \begin{cases} x & \text{if } y = 0x \text{ and } x \in \mathit{Correct}(N) \\ P(x) & \text{if } y = 1x \\ x_0 & \text{otherwise} \end{cases}$$

where x_0 is a fixed element from L. Because C is provably closed under reductions in L, there exists a proof system P'' for L 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, it follows that $\mathsf{C}(P'') = \mathsf{C}$.

For the proof of the converse implication, let P be a proof system for L in which every language $A \in \mathsf{C}$ is representable. We construct a complete set A_{C} for C by specifying a C-machine M_A accepting A_{C}. Elements of A_{C} are of the form

$$\langle x, N, 0^m, 0^n \rangle \ .$$

On such inputs M_A performs the following operations. M_A guesses a string $\pi \in \Sigma^{\leq n}$ and checks whether $P(\pi) = \mathit{corr}(x, N, 0^m)$. At this point we need that C can use nondeterminism. If this test fails, then M_A rejects the input. Otherwise, M_A simulates the machine N on input x for m steps and answers according to the answer obtained in this simulation. If the simulation does not terminate in m steps, M_A rejects the input.

Because C can use nondeterminism and has a universal machine, M_A is a C-machine and hence $A_{\mathsf{C}} \in \mathsf{C}$. To verify the hardness of A_{C} for C, let A be a language from C. Because A is representable in P, there exists a C-machine N accepting A such that $P \vdash_* \mathit{Correct}(N)$. Let p and q be polynomials bounding the running time of N and the proof size of $\mathit{Correct}(N)$ in P, respectively. Then the polynomial-time computable function

$$x \mapsto \langle x, N, 0^{p(|x|)}, 0^{q(|\mathit{corr}(x, N, 0^{p(|x|)})|)} \rangle$$

many-one reduces A to A_{C}. □

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.*

Proof. For the forward direction, assume that C has a many-one complete language (or function). It follows from Theorem 13 that C has a uniform enumeration $(N_i)_{i \in \mathbb{N}}$ of C-machines with polynomial running times p_{N_i}. Let G be a Turing machine generating the codes of the machines $N_1, N_2, \ldots$

We say that a string $v \in \Sigma^*$ is in *good form* if

$$v = \langle w_G, x, 0^{p_{N_i}(|x|)} \rangle$$

where $x \in \Sigma^*$ and w_G is a computation of the machine G eventually producing the code of the C-machine N_i. Apparently, we can check in polynomial time whether a given string is in good form.

We define $g : \Sigma^* \to L$ in the following way. If $v = \langle w_G, x, 0^{p_{N_i}(|x|)} \rangle$ is in good form, then $g(v) = corr(x, N_i, 0^{p_{N_i}(|x|)})$ (where N_i is the machine produced by G during the computation w_G), otherwise $g(v) = x_0$, where x_0 is a certain fixed string from L.

This polynomial-time computable function can be extended to a proof system P for L in which C is p-representable. Let P' be any proof system for L (L is recursively enumerable). We define the proof system P as follows:

$$P(y) = \begin{cases} g(v) & \text{if } y = 0v \\ P'(v) & \text{if } y = 1v. \end{cases}$$

Let A be any language from C. There exists a machine N_i from the uniform enumeration of the class C such that $L(N_i) = A$, and p_{N_i} is its polynomial-time bound. Let w_G be the computation of G producing the code of the machine N_i. The function $\lambda(x) = 0\langle w_G, x, 0^{p_{N_i}(|x|)} \rangle$ produces P-proofs of $Correct(N_i)$ in polynomial-time in $|x|$.

For the converse direction, assume that there exists a proof system P for L such that C is p-representable in it. Let R be the promise condition for C. Consider the language

$$A_{\mathsf{C}} = \{\langle x, N, 0^{p_N(|x|)}, w \rangle \mid x \in L(N),\ P(w) = corr(x, N, 0^{p_N(|x|)})\}$$

where N is a nondeterministic machine with polynomial time bound p_N and w is a P-proof of $corr(x, N, 0^{p_N(|x|)})$. We claim that A_{C} is the desired C-complete language.

Let us first argue that $A_{\mathsf{C}} \in \mathsf{C}$. We say that a string v is in good form if and only if $v = \langle x, N, 0^{p_N(|x|)}, w \rangle$ and $P(w) = corr(x, N, 0^{p_N(|x|)})$, where x, N, p_N, and w mean the same as above. Let us notice that if v is in good form, then $corr(x, N, 0^{p_N(|x|)}) \in L$. From the correctness condition it follows that N obeys promise R on input x.

There exists a polynomial-time Turing machine V which verifies whether a given input string v is in good form. By our general assumption on promise classes we have also a universal machine U with respect to the promise R. Using the machines V and U we can construct an R-machine K accepting A_{C}. The machine K, first runs V on input v, checking if v is in good form. If this test fails, then K rejects v, otherwise K runs U on input $\langle N, x, 0^{p_N(|x|)} \rangle$. Since U is a universal machine for C, the machine K is an R-machine.

To prove hardness of A_{C} for C, let A be any language in C. Since A is p-representable in P, there exists a C-machine N such that $L(N) = A$, and the P-proofs of $Correct(N)$ can be constructed in polynomial time in $|x|$. The function $f : \Sigma^* \to \Sigma^*$ defined by

$$f(x) = \langle x, N, 0^{p_N(|x|)}, w \rangle$$

performs a polynomial-time many-one reduction from A to A_{C}. The string w from the definition of f is the P-proof of $corr(x, N, 0^{p_N(|x|)})$. □

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 NP ∩ coNP is expressible in QBF (cf. [2, 14, 19, 21]). 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.*

Proof. It suffices to show that C contains a many-one complete problem if and only if C contains a $\leq$-complete problem. The forward implication is clear as $\leq$ is refined by many-one reductions.

Conversely, if C has a $\leq$-complete problem and C is provably closed in L under $\leq$, then we obtain a proof system P for L with $\mathsf{C}(P) = \mathsf{C}$ as in the proof of the forward implication of Theorem 14. Now the converse implication of Theorem 14 together with the assumption that C can use nondeterminism gives us a many-one complete problem for C. □

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. [9, 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. The proof is structured into the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$ and $2 \Rightarrow 4 \Rightarrow 1$.

For the proof of the direction $1 \Rightarrow 2$, let P be a p-optimal proof system for L and let A be a language from C. We choose a C-machine N accepting A. Because C is expressible in L, the set $\mathit{Correct}(N)$ is a polynomial-time decidable subset of L. Therefore we can devise a proof system P_A with polynomial-size proofs of $\mathit{Correct}(N)$ as follows:

$$P_A(y) = \begin{cases} x & \text{if } y = 0x \text{ and } x \in \mathit{Correct}(N) \\ P(x) & \text{if } y = 1x \\ x_0 & \text{otherwise} \end{cases}$$

where x_0 is a fixed element from L. Apparently, P_A is a proof system for L such that P_A-proofs of $\mathit{Correct}(N)$ can be constructed in polynomial time, i.e., A is p-representable in P_A. By the p-optimality of P we infer $P_A \leq_p P$. Thus P_A-proofs of $\mathit{Correct}(N)$ can be efficiently translated into P-proofs of these formulas and therefore A is also p-representable in P.

Implication $2 \Rightarrow 3$ holds because by Lemma 3, the P-easy subsets of L are expressible in L.

For the proof of $3 \Rightarrow 1$ we show that item 3 yields a recursive P-presentation of all P-easy subsets of L. As both Theorem 8 and Theorem 18 use the assumption of expressibility of $PS(L)$ in L, we obtain a p-optimal proof system for L by Theorem 8.

For this let P be a proof system for L in which all P-easy subsets of L are p-representable. Let $(t_i)_{i\in\mathbb{N}}$ be an enumeration of all deterministic polynomial-time clocked Turing transducers and let $(N_i, q_i)_{i\in\mathbb{N}}$ be an enumeration of all deterministic polynomial-time clocked Turing machines with their respective running times. The machines N_i will be candidates for machines accepting P-easy subsets of L where the promise condition R_{easy} for N_i is $L(N_i) \subseteq L$. Finally, let $\langle\cdot,\cdot\rangle$ be a polynomial-time computable and polynomial-time invertible bijective pairing function on $\mathbb{N}$.

Now, for $i \in \mathbb{N}$, consider the following set of algorithms M_i:

```
1  Input: x
2  compute numbers j and k with i = ⟨j,k⟩
3  compute N_j, q_j, and t_k
4  IF P(t_k(x)) ≠ corr(x, N_j, 0^{q_j(|x|)}) THEN reject
5  ELSE
6    IF N_j(x) accepts in ≤ q_j(|x|) steps THEN accept
7    ELSE reject
```

Apparently, these algorithms M_i can be computed in deterministic polynomial time. Further, each M_i only accepts inputs from L because if M_i accepts x, then we have an f-proof for $\mathit{corr}(x, N_j, 0^{q_j(|x|)})$, confirming that N_j obeys promise R_{easy} on x, i.e., $x \in L$.

On the other hand, each P-easy subset $A \subseteq L$ is p-representable in P with respect to some machine N_j and some transducer t_k computing the P-proofs

of $corr(x, N_j, 0^{q_j(|x|)})$ from input x. For $i = \langle j, k\rangle$ we then have $L(M_i) = A$. Thence M_i is a recursive P-presentation of all P-easy subsets of L.

For the direction $2 \Rightarrow 4$, let C be a promise class which is expressible in L. By item 2 we have a proof system P for L in which every language $A \in \mathsf{C}$ is p-representable. We will construct a complete set A_{C} for C as follows. If C is a language class, then A_{C} will be a many-one complete language for C, and if C is a function promise class, then A_{C} will be a many-one complete function for C. Elements of A_{C} are of the form

$$\langle x, N, 0^m, M, 0^n\rangle \ .$$

On such inputs the C-machine M_A for A_{C} performs the following operations. M_A computes in polynomial time the string $corr(x, N, 0^m)$. It then simulates the Turing transducer M on input x for at most n steps. If the simulation does not terminate, then M_A rejects (in case of a language class C) or outputs a fixed element (for a function class C).

Otherwise, let π be the output of the simulation $M(x)$. Next M_A checks whether $P(\pi) = corr(x, N, 0^m)$. If this test fails, then again M_A rejects the input (or outputs some fixed element). Otherwise, M_A simulates the machine N on input x for m steps and answers according to the answer obtained in this simulation.

By our general assumption, C has a universal machine and can perform polynomial-time computations. Therefore M_A is a C-machine and thus $A_{\mathsf{C}} \in \mathsf{C}$. To verify the hardness of A_{C} for C, let A be a language (or function) from C. Because A is p-representable in P, there exists a C-machine N accepting A such that P-proofs of $Correct(N)$ can be efficiently constructed. Let p be the running time of N and let M be a Turing transducer with polynomial running time q that computes P-proofs of $Correct(N)$ from input x. Then the polynomial-time computable function

$$x \mapsto \langle x, N, 0^{p(|x|)}, M, 0^{q(|x|)}\rangle$$

many-one reduces A to A_{C}.

For the final implication $4 \Rightarrow 1$ we need the assumption of the expressibility of $PS(L)$ in L. Because $PS(L)$ is a promise function class, item 4 together with this assumption 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.*

Proof. The proof of the direction $1 \Rightarrow 2$ proceeds similarly as the proof of $1 \Rightarrow 2$ in Theorem 18. Let P be an optimal proof system for L and let A be a language

from C. We choose a C-machine N accepting A. Because C is expressible in L, the set $\mathit{Correct}(N)$ is a polynomial-time subset of L. We construct a proof system P_A with polynomial-size proofs of $\mathit{Correct}(N)$ as follows:

$$P_A(y) = \begin{cases} x & \text{if } y = 0x \text{ and } x \in \mathit{Correct}(N) \\ P(x) & \text{if } y = 1x \\ x_0 & \text{otherwise} \end{cases}$$

where x_0 is a fixed element from L. Apparently, P_A is a proof system for L with $P_A \vdash_* \mathit{Correct}(N)$. By the optimality of P we infer $P_A \leq P$. Thus also $P \vdash_* \mathit{Correct}(N)$ holds, i.e., A is representable in P.

As in the previous proof, implication $2 \Rightarrow 3$ holds because the P-easy subsets of L are expressible in L by Lemma 3. The remaining implication $3 \Rightarrow 1$ follows by an analogous argument as in the proof of $3 \Rightarrow 1$ in Theorem 18. □

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

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 Complete Problems under 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 [7] 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 [7] 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. We choose a polynomial-time computable tupling function $\langle \cdot, \ldots, \cdot \rangle$ which is length injective, i.e., for all strings $x_1, \ldots, x_n, y_1, \ldots, y_n$, if $|\langle x_1, \ldots, x_n \rangle| = |\langle y_1, \ldots, y_n \rangle|$, then $|x_i| = |y_i|$ for $i = 1, \ldots, n$. 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. On 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. [7, 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 of the conference version of this paper for helpful comments and detailed suggestions on how to improve the 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. *Theoretical Computer Science*. To appear.
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. O. Beyersdorff and Z. Sadowski. Characterizing the existence of optimal proof systems and complete sets for promise classes. In *Proc. 4th International Computer Science Symposium in Russia*, volume 5675 of *Lecture Notes in Computer Science*, pages 47 – 58. Springer-Verlag, Berlin Heidelberg, 2009.
6. S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proc. 7th Annual ACM Symposium on Theory of Computing*, pages 83–97, 1975.
7. 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.
8. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
9. C. Glaßer, A. L. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. *Information and Computation*, 200(2):247–267, 2005.
10. C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.
11. 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.
12. 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.
13. J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.
14. J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.
15. J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1993.
16. 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.
17. 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.
18. J. Krajíček and P. Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.
19. 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.
20. Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets of TAUT. *Theoretical Computer Science*, 288(1):181–193, 2002.

21. Z. Sadowski. Optimal proof systems and complete languages. In *Local Proc. 4th Conference on Computability in Europe*, pages 407–414. University of Athens, 2008.
22. A. L. Selman. Much ado about functions. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, pages 198–212, 1996.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de