Optimal proof systems imply complete sets for promise classes*

JOHANNES KÖBLER, JOCHEN MESSNER, JACOBO TORÁN

Institut für Informatik
Humboldt-Universität zu Berlin
10099 Berlin, Germany
koebler@informatik.hu-berlin.de

Abt. Theoretische Informatik
Universität Ulm
89069 Ulm, Germany
messner,toran@informatik.uni-ulm.de

July 4, 2001

Abstract

A polynomial time computable function $h : \Sigma^* \to \Sigma^*$ whose range is a set L is called a proof system for L. In this setting, an h-proof for $x \in L$ is just a string w with $h(w) = x$. Cook and Reckhow defined this concept in [11] and in order to compare the relative strength of different proof systems for the set TAUT of tautologies in propositional logic, they considered the notion of p-simulation. Intuitively, a proof system h' p-simulates h if any h-proof w can be translated in polynomial time into an h'-proof w' for $h(w)$. Krajíček and Pudlák [18] considered the related notion of simulation between proof systems where it is only required that for any h-proof w there exists an h'-proof w' whose size is polynomially bounded in the size of w.

A proof system is called (p-)optimal for a set L if it (p-)simulates every other proof system for L. The question whether p-optimal or optimal proof systems for TAUT exist is an important one in the field. In this paper we show a close connection between the existence of (p-)optimal proof systems and the existence of complete problems for certain promise complexity classes like $\mathcal{UP}$, $\mathcal{NP} \cap$ *Sparse*, $\mathcal{RP}$ or $\mathcal{BPP}$. For this we introduce the notion of a test set for a promise class $\mathcal{C}$ and prove that $\mathcal{C}$ has a many-one complete set if and only if $\mathcal{C}$ has a test set T with a p-optimal proof system. If in addition the machines defining a promise class have a certain ability to guess proofs, then the existence of a p-optimal proof system for T can be replaced by the presumably weaker assumption that T has an optimal proof system.

Strengthening a result from Krajíček and Pudlák [18], we also give sufficient conditions for the existence of optimal and p-optimal proof systems.

*Results included in this paper have appeared in the conferences STACS'98 [19] and CCC'98 [14].

1 Introduction

A systematic study of the (proof length) complexity of different proof systems for Propositional Logic was started some time ago by Cook and Reckhow in [11]. There they defined the abstract notion of a proof system in the following way.

Definition 1 *Let $L \subseteq \Sigma^*$. A* proof system *for L is a (possibly partial) polynomial time computable function $h : \Sigma^* \to \Sigma^*$ whose range is L*[1]*. A string w with $h(w) = x$ is called an h-*proof *for x.*

Observe that a proof system h need not be polynomially honest since the shortest proof for $x \in L$ might be much longer than x.

Example 2 *The function h defined as*

$$h(w) = \begin{cases} \varphi & \text{if } w = \langle \varphi, v \rangle \text{ and } v \text{ is a resolution refutation of } \overline{\varphi}, \\ \text{undef.} & \text{otherwise} \end{cases}$$

is a proof system for the co-$\mathcal{NP}$ complete set TAUT of all propositional tautologies in disjunctive normal form.

Following [11], a *polynomially bounded proof system h* for TAUT is a proof system in which every tautology has a short proof. More formally, there is a polynomial q such that for every $\varphi \in$ TAUT, there is a string w of length bounded by $q(|\varphi|)$ with $h(w) = \varphi$. Many concrete proof systems for TAUT have been shown not to be polynomially bounded (see for example [25]). Besides the interest that concrete proof systems like, for example, resolution or Frege systems have in their own, a main motivation for the study of proof systems comes in fact from the following relation between the $\mathcal{NP}$ versus *co-*$\mathcal{NP}$ question and the existence of polynomially bounded systems.

Theorem 3 [11] *$\mathcal{NP} =$ co-$\mathcal{NP}$ if and only if a polynomially bounded proof system for TAUT exists.*

This result was the motivation for the so called *Cook-Reckhow Program*: a way to prove that $\mathcal{NP}$ is different from *co-*$\mathcal{NP}$ might be to study more and more powerful proof systems, showing that they are not polynomially bounded, until hopefully we have gained enough knowledge to be able to separate $\mathcal{NP}$ from *co-*$\mathcal{NP}$ (see [8]).

In order to compare the relative power of different proof systems, the notion of p-simulation was introduced in [11]. We also consider the presumably weaker notion of simulation studied in [18].

[1] The original definition allows in fact the use of different alphabets for the domain and range of h, but for the purposes of this paper the given definition suffices.

Definition 4 *Let h and h' be proof systems for a language L. We say that h* simulates *h' if there is a polynomial p and a function $\gamma : \Sigma^* \to \Sigma^*$ with $|\gamma(x)| \leq p(|x|)$ for every $x \in \Sigma^*$ such that γ translates h'-proofs into h-proofs. In other words, for every $x \in L$ and every h'-proof w of x, $\gamma(w)$ is an h-proof of x. If additionally $\gamma \in \mathcal{FP}$, we say h* p-simulates *h'.*

It is easy to see that simulation and p-simulation are preorders, i.e. reflexive and transitive relations. It is also clear that if a proof system h which is not polynomially bounded simulates another proof system h', then h' cannot be polynomially bounded. Cook and Reckhow used p-simulation in order to classify different proof systems for TAUT according to their derivation strength.

The notion of (p-)simulation between proof systems is closely related to the notion of reducibility between decision problems. Continuing with this analogy, the notion of a complete problem corresponds to the notion of an optimal proof system.

Definition 5 (cf. [18, 17, 8]) *A proof system for a language L is optimal (p-optimal) if it simulates (p-simulates) every proof system for L.*

An important open problem (posed in [18, 8]) is whether an optimal or even p-optimal proof system for TAUT exists. Observe that if this were the case, then in order to separate $\mathcal{NP}$ from *co-*$\mathcal{NP}$ it would suffice to prove that some specific proof system is not polynomially bounded.

We show that the assumption that there is a (p-)optimal proof system for certain languages is closely related to the existence of complete problems for certain promise classes. The connection between the existence of (p-)optimal proof systems and the existence of many-one complete sets is formalized by introducing the concept of test sets. Roughly speaking, a test set allows us to verify that a given nondeterministic polynomial-time machine behaves well (i.e., in accordance with the promise) on a given input x. Hence, in some sense, the complexity of a promise is represented by the complexity of its test sets. We then obtain in a master theorem that a promise class $\mathcal{C}$ has a many-one complete set if and only if $\mathcal{C}$ has a test set with a p-optimal proof system.

As the classes $\mathcal{UP}$, *Few*$\mathcal{P}$, *Few*, and $\mathcal{NP} \cap$ *Sparse* have test sets in *co-*$\mathcal{NP}$ this in turn implies that a p-optimal proof system for TAUT suffices to obtain many-one complete sets for these classes. We also show that the probabilistic classes $\mathcal{BPP}$, $\mathcal{RP}$, $\mathcal{ZPP}$, and $\mathcal{MA}$ have test sets in Π_2^p as well as in Σ_2^p, and that $\mathcal{AM}$ has test sets in Π_3^p and in Σ_3^p. Hence, a many-one complete set for $\mathcal{BPP}$, $\mathcal{RP}$, $\mathcal{ZPP}$, and $\mathcal{MA}$ (resp. $\mathcal{AM}$) is implied by a p-optimal proof system for TAUT_2 or for SAT_2 (resp. for TAUT_3 or for SAT_3). We also show that $\mathcal{NP} \cap$ *co-*$\mathcal{NP}$ has a test set reducible to SAT $\times$ TAUT which allows us to improve the main result of [22] by showing that already the existence of p-optimal proof systems for TAUT and for SAT suffices to obtain a complete problem for $\mathcal{NP} \cap$ *co-*$\mathcal{NP}$. If in addition the machines defining a promise class have a certain ability to guess proofs (see the definition of $\mathcal{NP}$-assertions in Definiton 14), then it suffices to assume that the $\mathcal{C}$ has a test set with an optimal proof system. Under the mentioned classes this holds for $\mathcal{NP} \cap$ *Sparse*, for $\mathcal{MA}$, and for $\mathcal{AM}$.

These results strengthen the intuitive connection between the notions of optimal proof systems and complete sets. At the same time, they give some evidence that (p-)optimal proof systems might not exist for the considered logical languages since many-one complete problems for these classes have been searched for without success.

In [21] it was observed that the class of disjoint $\mathcal{NP}$-pairs has a complete pair if TAUT has an optimal proof system. A pair (A, B) of $\mathcal{NP}$-languages A, B belongs to this class when $A \cap B = \emptyset$. However, in [21] a somewhat weak form of many-one reducibility is used which is only concerned with inputs from $A \cup B$. Formally, in [21] a pair (A, B) is said to many-one reduce to a pair (C, D) if for some $f \in \mathcal{FP}$, $f(A) \subseteq C$ and $f(B) \subseteq D$. By generalizing our approach to function classes we can use the assumption that TAUT has an optimal proof system to conclude that the class of disjoint $\mathcal{NP}$-pairs has a complete pair with respect to the following stronger notion of many-one reducibility: (A, B) strongly many-one reduces to (C, D) if for some $f \in \mathcal{FP}$, $f^{-1}(C) = A$ and $f^{-1}(D) = B$.

The before-mentioned set of results can be interpreted as the study of necessary conditions for the existence of (p-)optimal proof systems. We consider in Section 6 also sufficient conditions for the existence of this kind of proof systems. The following sufficient conditions have been proved by Krajíček and Pudlák.

Theorem 6 [18]

If $\mathcal{NE} =$ co-$\mathcal{NE}$ then optimal proof systems for TAUT exist.

If $\mathcal{E} = \mathcal{NE}$ then p-optimal proof systems for TAUT exist.

We improve this result by weakening the conditions that are sufficient for the existence of optimal and p-optimal proof systems for TAUT. We show that if the deterministic and nondeterministic double exponential time complexity classes coincide ($\mathcal{EE} = \mathcal{NEE}$) then p-optimal proof systems for TAUT exist, and that $\mathcal{NEE} =$ *co*-$\mathcal{NEE}$ is sufficient for the existence of optimal proof systems for TAUT. In fact we give a probably weaker sufficient condition showing that a collapse of the class of tally sets in nondeterministic double exponential time to the deteterministic counterpart suffices for the existence of optimal proof systems for TAUT.

By the relationships between optimal proof systems and complete sets, one would expect that optimal proof systems for sets like TAUT do not exist. The sufficient conditions show however that it would be very hard to prove that optimal proof systems do not exist, since this would imply a separation of complexity classes.

The rest of this article is structured as follows. In the next section we give some preliminaries and study some closure properties of the class of languages that have a (p-)optimal proof system. These results are interesting on their own but mainly serve as a technical tool for the following sections. In Section 3 we provide the setting needed to formalize the informal notion of a promise class. Further, we present two master theorems that are applied in Section 4 to obtain the completeness consequences mentioned above (where we also consider nonuniform reducibilities). In Section 5 we briefly discuss how these ideas can be adjusted for the case of promise function classes. As an application, we obtain the already mentioned completeness consequence for the class of disjoint $\mathcal{NP}$-pairs. Finally, in Section 6 we consider sufficient conditions for the existence of (p)-optimal proof systems.

2 Preliminaries

Let QBF be the set of all valid quantified Boolean formulas

$$(Q_1 x_1) \cdots (Q_n x_n)\, F(x_1, \ldots, x_n),$$

where $F(x_1, \ldots, x_n)$ is a Boolean formula over the variables $x_1, \ldots, x_n$ and each Q_i is either $\exists$ or $\forall$. By TAUT_k we denote the set of all QBF formulas with at most $k-1$ quantifier alternations ($\exists$ followed by $\forall$, or $\forall$ followed by $\exists$) starting with $\forall$. Similarly, SAT_k denotes the set of all QBF formulas with at most $k-1$ quantifier alternations starting with $\exists$. As usual, in the case of $k = 1$ we omit the index and simply write SAT for SAT_1 and TAUT for TAUT_1.

We assume some familiarity with complexity theory and refer the reader to [2, 3] for standard notions and for the definition of complexity classes. A language A many-one reduces to a language B (in symbols: $A \leq_m^p B$) if there is a polynomial-time computable function f (in symbols: $f \in \mathcal{FP}$) such that for all strings x, $x \in A$ if and only if $f(x) \in B$.

We use Turing machines as our basic computational model. In particular, we consider clocked deterministic and nondeterministic polynomial-time Turing machines (PTM and NPTM for short). To represent these machines we use an encoding which allows us to obtain from the machine code N easily a polynomial p_N that bounds the running time of the machine (in Section 3 we will consider some further restrictions on the encodings of NPTMs and PTMs). In the sequel, we will not distinguish between a machine and its code.

We consider only languages over the alphabet $\Sigma = \{0, 1\}$ (this means that problem instances as, e.g., Boolean formulas have to be suitably encoded). By Σ^* we denote the set of all binary strings, and by $\Sigma^{\leq n}$ the set of strings of length at most n. Sometimes we identify a string $w \in \Sigma^*$ with the positive integer that has $1w$ as binary representation. A set T is called *tally* (in symbols: $T \in \mathit{Tally}$) if T is a subset of $\{0^n \mid n \geq 0\}$; a set S is called *sparse* (in symbols:

$T \in \mathit{Sparse}$) if there is a polynomial $p(n)$ that bounds the cardinality of $S \cap \Sigma^{\leq n}$. A *tupling function* maps tuples of words to single words. It is injective and can be computed and inverted in polynomial time. If not specified otherwise, we assume some standard tupling function denoted by $\langle \cdot, \cdots, \cdot \rangle$. We use the special value *undef.* to indicate that a partial function is undefined on x (clearly if $g(x) =$ *undef.* then also $f(g(x)) =$ *undef.*).

In the rest of this section we investigate closure properties of the class of all languages which have a (p-)optimal proof system. Since, as we show, this class is closed under $\leq_m^p$ and under join, it follows [6] that it is a promise class in the classical sense of the tree-structure or of the leaf-language approach (in fact, in [6] it is shown that both approaches are equivalent and that any class closed under $\leq_m^p$ and under join is a promise class in that sense).

All observations made in this section refer to the notions of optimality and p-optimality interchangeably. We only give the proofs for the p-optimal case since they can easily be adapted to the case of optimality.

Lemma 7 *If A has a (p-)optimal proof system and if $B \leq_m^p A$, then B has a (p-)optimal proof system, too.*

Proof: Let h be a p-optimal proof system for A and let B many-one reduce to A via $f \in \mathcal{FP}$. Then h' defined by

$$h'(\langle x, w \rangle) = \begin{cases} x & h(w) = f(x), \\ \text{undef.} & \text{otherwise} \end{cases}$$

is certainly a proof system for B. To show that h' is p-optimal, let g' be a proof system for B. By setting $g(0w) = h(w)$ and $g(1w) = f(g'(w))$ we obtain a proof system g for A. Since h is p-optimal, there is a function $t \in \mathcal{FP}$ translating g-proofs to h-proofs implying that

$$h(t(1w)) = g(1w) = f(g'(w)).$$

This implies

$$h'(\langle g'(w), t(1w) \rangle) = g'(w).$$

Hence, h' p-simulates g'. ∎

The *join* $A \oplus B$ of two languages is given by $0A \cup 1B$. It is a least upper bound for A and B with respect to the ordering induced by $\leq_m^p$. The *direct product* of A and B is given by $A \times B = \{\langle a, b \rangle \mid a \in A,\, b \in B\}$. Clearly, if a class is closed under $\leq_m^p$, then closure under intersection implies closure under direct product, as follows from the equality $A \times B = (A \times \Sigma^*) \cap (\Sigma^* \times B)$. Closure under direct product in turn implies closure under join, as $A \oplus B \leq_m^p A \times B$ for nonempty A and B.

Lemma 8 *If A and B have (p-)optimal proof systems, then so have $A \times B$, $A \oplus B$, and $A \cap B$.*

Proof: As observed above it suffices to consider $A \cap B$. Let h_1 and h_2 be p-optimal proof systems for A and B, respectively. A p-optimal proof system for $A \cap B$ is given by the $\mathcal{FP}$ function

$$h : w \mapsto \begin{cases} x & \text{if } w = \langle u, v \rangle, \text{ and } x = h_1(u) = h_2(v), \\ \text{undef.} & \text{otherwise.} \end{cases}$$

Clearly, h is a proof system for $A \cap B$. To show that h is p-optimal, let f be some proof system for $A \cap B$. By setting $f_i(1w) = f(w)$ and $f_i(0w) = h_i(w)$ $(i = 1, 2)$, f can be extended to proof systems f_1 and f_2 for A and B, respectively. Let $t_i \in \mathcal{FP}$ be a function translating f_i-proofs to h_i-proofs. Then the function $t(w) = \langle t_1(1w), t_2(1w) \rangle$ translates f-proofs to h-proofs, i.e., $h(t(w)) = f(w)$, as $f(w) = f_i(1w) = h_i(t_i(1w))$, for $i = 1, 2$. ▮

The proof of the next lemma is straightforward and is therefore omitted.

Lemma 9

Any set in $\mathcal{P}$ has a p-optimal proof system.

Any set in $\mathcal{NP}$ has an optimal proof system.

It is an open question whether sets with a (p-)optimal proof system exist outside of $\mathcal{NP}$ (respectively $\mathcal{P}$).

3 Complete sets for promise classes

In this section we provide a quite general approach to promise classes. In the classical leaf-language or tree-structure approach [7, 27, 6], promises are restricted to be predicates on computation trees (respectively leaf strings) of nondeterministic polynomial-time Turing machines. However it does not seem to be an easy task to define a natural promise for the $\leq_m^p$-closure of $\mathcal{NP} \cap \mathit{Sparse}$ (in spite of the fact that results in [6] show that it is actually a promise class, as it is closed under $\leq_m^p$ and under join).

In order to state our results as general as possible, we allow promises to be formulated in a quite unrestricted way. In this paper, a *promise* R is described by a predicate on the set of all pairs consisting of an NPTM N and an input string x, i.e., $R(N, x)$ means that N obeys promise R on input x. We call N an *R-machine* if N obeys R on any input $x \in \Sigma^*$. In the sequel, we will only allow promises R for which at least one R-machine exists. The *acceptance criterion* is also a binary predicate Q on NPTMs and strings. The language accepted by the NPTM N (when applying the acceptance criterion Q) is given by

$$L_Q(N) = \{x \in \Sigma^* \mid Q(N, x)\}.$$

Finally, for a promise predicate R and an acceptance criterion Q, we define the promise class

$$\mathcal{C}_{Q,R} = \{L_Q(N) \mid N \text{ is an } R\text{-machine}\}$$

and call (Q, R) a *defining pair* for $\mathcal{C}_{Q,R}$.

Definition 10 *A class of languages $\mathcal{C}$ is called a promise class if $\mathcal{C} = \mathcal{C}_{Q,R}$ for some promise predicate R and acceptance criterion Q.*

Notice that a promise class $\mathcal{C}_{Q,R}$ cannot be empty as we assume that some R-machine exists. In order to obtain completeness results, this setting is still too unrestricted. In fact, for any nonempty countable class $\mathcal{L}$ of languages one can define Q and R such that $\mathcal{L} = \mathcal{C}_{Q,R}$. Therefore we often restrict our consideration to pairs (Q, R) that fulfill the following two conditions A1 and A2. Basically, A1 demands the existence of a universal NPTM (with respect to Q and R) and A2 requires that $\mathcal{C}_{Q,R}$ is closed under many-one reducibility (in a constructive way).

A1: There is an NPTM U, and a tupling function $\langle \cdot, \cdot, \cdot \rangle$ such that the following two conditions hold for all NPTMs N, all $x \in \Sigma^*$, and all $s \geq p_N(|x|)$.

1. $Q(N, x) \Longleftrightarrow Q(U, \langle N, x, 0^s \rangle)$.
2. $R(N, x) \implies R(U, \langle N, x, 0^s \rangle)$.

A2: There is a binary operation $\circ$ mapping an NPTM N and a polynomial-time transducer M to an NPTM $N \circ M$ such that the following three conditions hold for all NPTMs N, all PTMs M, and all $x \in \Sigma^*$ (f_M denotes the function computed by M).

1. $Q(N, f_M(x)) \Longleftrightarrow Q(N \circ M, x)$.
2. $R(N, f_M(x)) \implies R(N \circ M, x)$.
3. The set $\{N \circ M' \mid M' \text{ is a polynomial-time transducer}\}$ is recursively enumerable.

It is easy to verify that the conditions A1 and A2 hold in the leaf-language as well as in the tree-structure approach.

A natural defining pair for $\mathcal{UP}$ is (Q, R) where $R(N, x)$ holds if N is a NPTM with at most one accepting path on input x and $Q(N, x)$ is true if N has at least one accepting path on input x. To show that A1 holds, let U be a nondeterministic universal Turing machine that on input $\langle N, x, 0^s \rangle$ simulates s steps of machine N

on input x. Then it is clear that for a standard encoding of NPTMs, U works in polynomial time and fulfills A1. Further, A2 also holds by defining $N \circ M$ to be the machine that on input x computes $M(x)$ and then simulates N on $M(x)$ (of course, the attached polynomial time-bounds have to be adjusted appropriately).

Proposition 11 *The class $\mathcal{UP}$ has a defining pair which fulfills A1 and A2.*

We will see in Section 4 how other promise classes like $\mathcal{NP} \cap$ *co-*$\mathcal{NP}$, *Few*, *Few*$\mathcal{P}$, $\mathcal{BPP}$, $\mathcal{RP}$, $\mathcal{ZPP}$, $\mathcal{AM}$ and $\mathcal{MA}$ can be characterized in a natural way by defining corresponding pairs (Q, R) which fulfill A1 and A2.

Next we introduce the concept of a test set which is central to our approach. The complexity of a test set serves to some extend as a measure for the complexity inherent to a defining pair.

Definition 12 *Let $\mathcal{C}$ be a promise class, and let (Q, R) be a defining pair for $\mathcal{C}$. Then a set $T \subseteq \Sigma^*$ is called a (Q, R)-*test set for *$\mathcal{C}$ if the following two conditions are fulfilled.*

- *If $\langle N, x, 0^s \rangle \in T$, then $s \geq p_N(|x|)$ and $R(N, x)$ holds.*
- *For any $L \in \mathcal{C}$ there is an NPTM N that accepts L, i.e., $L_Q(N) = L$, and that passes test T, i.e., there is a polynomial p such that for all inputs $x \in \Sigma^*$, $\langle N, x, 0^{p(|x|)} \rangle \in T$.*

Thus, any element $\langle N, x, 0^s \rangle$ belonging to a (Q, R)-test set T serves as an assertion that N behaves well (according to R) on input x. For example, we can use the generic test set

$$T_{(Q,R)} = \{\langle N, x, 0^s \rangle \mid R(N, x) \text{ and } s \geq p_N(|x|)\}$$

for a defining pair (Q, R). In the case of $\mathcal{UP}$, for example, one just has to verify that there is at most one accepting path; a simple task in *co-*$\mathcal{NP}$.

Proposition 13 *$\mathcal{UP}$ has a test set in co-$\mathcal{NP}$.*

Very informally, the intuitive idea behind the notion of a test set T is that we can obtain a complete language for $\mathcal{C}_{Q,R}$, provided that we can enable an R-machine to decide T (see the proofs of Theorems 16 and 19 for details). In order to make this intuition precise we need the following notion.

Definition 14 *Let $\mathcal{A}$ be a class of languages and (Q, R) be a defining pair for a promise class $\mathcal{C}$. We say that $\mathcal{A}$-*assertions are useful for *(Q, R), if for any language $B \in \mathcal{A}$ and any NPTM N the following holds: if N obeys R for any $x \in B$ then there is a language $C \in \mathcal{C}$ such that*

$$C \cap B = L_Q(N) \cap B.$$

Occasionally, when it is clear from the context which defining pair (Q, R) we associate with a promise class $\mathcal{C}$ then we just say that *$\mathcal{A}$-assertions are useful for $\mathcal{C}$*; similarly, we sometimes call a (Q, R)-test set simply a *test set for $\mathcal{C}$*.

The next lemma is needed in the proof of the main result of this section (Theorem 16).

Lemma 15 *$\mathcal{P}$-assertions are useful for defining pair (Q, R) that fulfills A2.*

Proof: Let $B \in \mathcal{P}$ and let N be an NPTM such that $R(N, x)$ holds for all $x \in B$. We can assume that B is nonempty (otherwise the statement is true as we assume that $\mathcal{C}_{Q,R}$ is nonempty). Let M be a PTM that computes the $\mathcal{FP}$ function f defined as $f(x) = x$, if $x \in B$, and $f(x) = y$ otherwise, where y is a fixed string in B. Since $R(N, f(x))$ holds for all x, we conclude by assumption A2.2 that $N' = N \circ M$ is an R-machine, implying that $C = L_Q(N') \in \mathcal{C}_{Q,R}$. By the definition of f and by A2.1 it follows that for all $x \in B$, $Q(N, x) \iff Q(N', x)$. Hence, $C \cap B = L_Q(N) \cap B$. ∎

Now we are ready to prove our main result, namely the equivalence *1* $\iff$ *2* of the next theorem. The equivalence *2* $\iff$ *3* has been observed already in [16] for $\mathcal{C} = \mathcal{NP} \cap co\text{-}\mathcal{NP}$ and in [13] for $\mathcal{C} = \mathcal{UP}$ and for $\mathcal{C} = \mathcal{BPP}$.

Theorem 16 *Let $\mathcal{C}$ be a promise class and let (Q, R) be a defining pair for $\mathcal{C}$ which fulfills A1 and A2. Then the following conditions are equivalent.*

1. *$\mathcal{C}$ has a (Q, R)-test set with a p-optimal proof system.*
2. *$\mathcal{C}$ has a many-one complete set.*
3. *There is a recursive enumeration $N_1, N_2, \ldots$ of R-machines such that $\mathcal{C} = \{L_Q(N_i) \mid i \geq 1\}$.*
4. *$\mathcal{C}$ has a (Q, R)-test set in $\mathcal{P}$.*

Proof: *1* $\Longrightarrow$ *2*. Let T be a (Q, R)-test set for $\mathcal{C}$ which has a p-optimal proof system h. Remember that for every fixed NPTM N that passes test T, there is a polynomial p such that the language

$$T_N = \{\langle N, x, 0^{p(|x|)}\rangle \mid x \in \Sigma^*\}$$

is a subset of T. Hence it follows that there is a proof system g for T with the property that for all $x \in \Sigma^*$,

$$g(1x) = \langle N, x, 0^{p(|x|)}\rangle.$$

Since h is a p-optimal proof system for T, there is a function $t \in \mathcal{FP}$ such that for every $x \in \Sigma^*$, $h(t(1x)) = \langle N, x, 0^{p(|x|)}\rangle$. Thus, $L_Q(N)$ is easily seen to reduce to the set

$$A = \{\langle N', x, 0^s, w\rangle \mid x \in L_Q(N') \wedge h(w) = \langle N', x, 0^s\rangle\}$$

via the reduction

$$f_N : x \mapsto \langle N, x, 0^{p(|x|)}, t(1x)\rangle .$$

Now, let $B = \{\langle N', x, 0^s, w\rangle \mid h(w) = \langle N', x, 0^s\rangle\}$. Notice that the reductions f_N defined above map only to elements in B. Therefore, any language C with the property that $A = C \cap B$ is hard for $\mathcal{C}$. We show that such a language C exists in the class $\mathcal{C}$. Let U be a universal NPTM according to A1 and let $U' = U \circ M$, where M is a transducer computing the projection that maps $\langle a, b, c, d\rangle$ to $\langle a, b, c\rangle$ where for the latter encoding the tupling function due to A1 is used. Observe that (by A1 and A2) U' obeys R for all $y \in B$ and that $Q(U', \langle N, x, 0^s, w\rangle) \iff Q(N, x)$. Since by Lemma 15, $\mathcal{P}$-assertions are useful for (Q, R), and since $B \in \mathcal{P}$, it follows that there is a language $C \in \mathcal{C}$ with the property that $C \cap B = A \cap B = A$.

2 $\Longrightarrow$ *3.* Let C be a many-one complete set for $\mathcal{C}$ and let N_C be an R-machine with $C = L_Q(N_C)$. Since C is complete for $\mathcal{C}$, any language L in $\mathcal{C}$ can be decided by an R-machine of the form $N_C \circ M$, where M is a polynomial-time transducer computing the reduction from L to C. (Notice that A2.2 implies that $N_C \circ M$ is an R-machine and that A2.1 implies that $L_Q(N_C \circ M) = L$). Thus, due to A2, the recursively enumerable set

$$S = \{N_C \circ M \mid M \text{ is a PTM}\}$$

has the properties required for condition *3*.

3 $\Longrightarrow$ *4.* Let M be a Turing machine that accepts the set $S = \{N_i \mid i \geq 1\}$ given by the recursive enumeration of *3*. It now suffices to observe that the set

$$T = \{\langle N, x, 0^s\rangle \mid p_N(|x|) \leq s \text{ and } M \text{ accepts } N \text{ in } \leq s \text{ steps}\}$$

is a (Q, R)-test set in $\mathcal{P}$.

4 $\Longrightarrow$ *1.* This implication follows immediately from Lemma 9. ∎

By combining Theorem 16 with Proposition 13 we get the following corollary.

Corollary 17 *If* TAUT *has a p-optimal proof system then $\mathcal{UP}$ has a many-one complete set.*

We notice that if a promise class fulfills A1 and A2 and has a complete set under polynomial time many-one reducibility, then it also has complete sets under less complex many-one reductions (like e.g. logspace-reductions). To see this, consider a direct proof of implication *4* $\Longrightarrow$ *2*: If T is a test set fulfilling *4*, then clearly the universal machine given by A1 obeys R on any $y \in T$. Hence, as $\mathcal{P}$-assertions are useful for $\mathcal{C}$, there is a set $C \in \mathcal{C}$ such that $C \cap T = L_Q(U) \cap T$. Now let $L \in \mathcal{C}$ and let N be an NPTM with $L_Q(N) = L$ that passes test T with polynomial q. Then the mapping $x \mapsto \langle N, x, 0^{q(|x|)}\rangle$ reduces L to C. Hence, the complexity of the reduction is basically that of computing the tupling function.

But the latter can be chosen to be very simple: if A1 holds with a universal machine U and a certain tupling function then we can use a polynomial time machine M that translates a very simple tuple-representation into this one and obtain a machine $U' = U \circ M$ that (using A2) fulfills the conditions of A1 with respect to the simple tuple-representation. In fact, all completeness consequences in this article carry over to many-one reducibilities that are simple to compute as, e.g., logspace-reducibility.

Next we derive completeness consequences from the assumption that the promise class under consideration has a test set with an optimal proof system. We obtain similar implications if the promise class can even use $\mathcal{NP}$-assertions (see Theorem 19). However, the following equivalence holds without this assumption.

Theorem 18 *Let $\mathcal{C}$ be a promise class and (Q,R) be a defining pair for $\mathcal{C}$. Then the following two conditions are equivalent.*

1. *$\mathcal{C}$ has a (Q,R)-test set with an optimal proof system.*
2. *$\mathcal{C}$ has a (Q,R)-test set in $\mathcal{NP}$.*

Proof: By Lemma 9, *2* implies *1*. For the opposite implication assume that we have a (Q,R)-test set T for $\mathcal{C}$ and an optimal proof system h for T. Let

$$T' = \{\langle N, x, 0^{s+t}\rangle \mid \exists w \in \Sigma^{\leq s}\colon\ h(w) = \langle N, x, 0^t\rangle\}\,.$$

Clearly, $T' \in \mathcal{NP}$. To show that T' is a (Q,R)-test set for $\mathcal{C}$, we prove that each machine N which passes T also passes T'. If N passes T, then there is some polynomial p bounding the running time of N and having the property that for all $x \in \Sigma^*$, $\langle N, x, 0^{p(|x|)}\rangle \in T$. It is easy to define a proof system g for $T_N = \{\langle N, x, 0^{p(|x|)}\rangle \mid x \in \Sigma^*\}$ such that for any $x \in \Sigma^*$, $g(1x) = \langle N, x, 0^{p(|x|)}\rangle$. As h simulates g, there is a polynomial q such that for all x, $\langle N, x, 0^{p(|x|)}\rangle$ has an h-proof of size $q(|x|)$. Thus for any x, $\langle N, x, 0^{p(|x|)+q(|x|)}\rangle \in T'$. But this means that N passes test T'. ∎

Theorem 19 *Let $\mathcal{C}$ be a promise class and let (Q,R) be a defining pair for $\mathcal{C}$ which fulfills A1. If $\mathcal{NP}$-assertions are useful for (Q,R), then 1 implies 2.*

1. *$\mathcal{C}$ has a (Q,R)-test set with an optimal proof system.*
2. *$\mathcal{C}$ has a many-one complete set.*

Proof: Let T' be a (Q,R)-test set for $\mathcal{C}$ which has an optimal proof system. By Theorem 18, there is a (Q,R)-test set T for $\mathcal{C}$ which is in $\mathcal{NP}$ (notice that $T \in \mathcal{NP}$ holds independently of the specific tupling function used to encode the tuples in T, hence we may assume that the tupling function due to A1 is used). Consider the set

$$A \;=\; \{\langle N, x, 0^s\rangle \in T \mid x \in L_Q(N)\}.$$

Notice that $A = L_Q(U) \cap T$ where U is a universal NPTM given by A1. Notice also that by A1.2, U obeys R on any $y \in T$. As $\mathcal{NP}$-assertions are useful for (Q, R), there is a language $C \in \mathcal{C}$ such that $C \cap T = L_Q(U) \cap T = A$. We now show that C is complete for $\mathcal{C}$. Let L be a set in $\mathcal{C}$ and let N be an NPTM with $L = L_Q(N)$ which passes test T with respect to a polynomial p. Then the mapping $x \mapsto \langle N, x, 0^{p(|x|)}\rangle$ reduces L to C (as well as to A). ∎

Notice that conditions 1 and 2 of Theorem 19 are equivalent if we additionally require that (Q, R) fulfills A2. This follows from the fact stated in Theorem 16 that the existence of a complete language implies the existence of a test set in $\mathcal{P}$.

Even if the promise class under consideration cannot use $\mathcal{NP}$-assertions we can still derive completeness consequences with respect to nonuniform reducibilities. In order to do so we define the concept of a length dependent test set.

Definition 20 *A test set T is called* length dependent *if $\langle N, x, 0^s\rangle \in T$ implies $\langle N, y, 0^s\rangle \in T$ for all inputs y of length $|y| = |x|$.*

It is clear that from any test set T for (Q, R) we can generically obtain a length dependent test set

$$T' \;=\; \{\langle N, x, 0^s\rangle \mid \forall y \in \Sigma^{|x|}, \langle N, y, 0^s\rangle \in T_{(Q,R)}\}$$

for (Q, R). Actually, in [22, 19] only length dependent test sets were (implicitly) used to derive completeness consequences. However notice that in order to obtain a length dependent test set an additional $\forall$-quantifier may be needed. However, if we apply this construction to a test set $T \in$ *co*-$\mathcal{NP}$, then also T' belongs to this class.

Proposition 21 *$\mathcal{UP}$ has a length dependent test set in co-$\mathcal{NP}$.*

A function $f \in \mathcal{FP}/poly$ with $f(x) \in B \iff x \in A$ is called a *nonuniform many-one reduction* from A to B.

Theorem 22 *Let $\mathcal{C}$ be a promise class and let (Q, R) be a defining pair for $\mathcal{C}$ that fulfills A1 and A2. Then* 1 *implies* 2.

1. *$\mathcal{C}$ has a length dependent (Q, R)-test set with an optimal proof system.*

2. *$\mathcal{C}$ has a complete set under nonuniform many-one reducibility.*

Proof: The proof follows the lines of the proof of *1* $\Longrightarrow$ *2* of Theorem 16. Let T be a length dependent (Q,R)-test set for $\mathcal{C}$ that has an optimal proof system h. Then for every fixed NPTM N that passes test T, there is a polynomial p such that the language

$$T_N = \{\langle N, 0^n, 0^{p(n)}\rangle \mid n \geq 0\}$$

is a subset of T. Hence, as T_N is easy to recognize, it follows that h-proofs for $\langle N, 0^n, 0^{p(n)}\rangle$ are short (i.e., their length is polynomially bounded in n). Thus, $L_Q(N)$ is easily seen to reduce to the set

$$A = \{\langle N', x, 0^s, w\rangle \mid x \in L_Q(N') \wedge h(w) = \langle N', 0^{|x|}, 0^s\rangle\}$$

via the reduction

$$f_N : x \mapsto \langle N, x, 0^{p(|x|)}, w\rangle,$$

where the h-proof w of $\langle N, 0^{|x|}, 0^{p(|x|)}\rangle$ is given as advice by f_N. Now, let $B = \{\langle N', x, 0^s, w\rangle \mid h(w) = \langle N', 0^{|x|}, 0^s\rangle\}$ and follow the rest of the proof of implication *1* $\Longrightarrow$ *2* of Theorem 16 that shows that there is a set $C \in \mathcal{C}$ with $C \cap B = A$ that is hard for $\mathcal{C}$ (here under nonuniform reducibility). ∎

By combining Theorem 22 with Proposition 21 we get the following corollary.

Corollary 23 *If* TAUT *has an optimal proof system then* $\mathcal{UP}$ *has a complete set under nonuniform many-one reducibility.*

4 Applications to other promise classes

Whereas in the last section $\mathcal{UP}$ served as our standard example for a promise class, we use in this section the assumption that certain languages have (p-)optimal proof systems to derive further completeness consequences for various other promise classes. We start by sketching how defining pairs (Q,R) (i.e. machine models) for promise classes like $\mathcal{NP} \cap co\text{-}\mathcal{NP}$, $\Sigma_k^p \cap \Pi_k^p$, $k \geq 2$, *Few*, *Few*$\mathcal{P}$, and $\mathcal{NP} \cap$ *Sparse* can be obtained.

A machine model for $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ can be obtained by combining two $\mathcal{NP}$-machines N_1 and N_2 which accept complementary languages into a machine N that in the first (nondeterministic) step, branches left to N_1 and right to N_2. So, for $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ the promise $R(N,x)$ states that on input x, either N_1 or N_2 accepts but not both. $Q(N,x)$ holds if N_1 has an accepting path on input x.

Machine models for $\Sigma_k^p \cap \Pi_k^p$ for each $k \geq 2$ can be obtained in a similar way by combining two Σ_k^p-machines which accept complementary languages (Σ_k^p-machines may be defined syntactically or by the tree-structure of an NPTM). So the promise $R(N,x)$ holds when N branches in the first step to two Σ_k^p-computations, where the left one is accepting if and only if the right one is rejecting. Here $Q(N,x)$ holds if N_1 accepts x in a Σ_k^p-way.

The classes *Few* and *Few*$\mathcal{P}$ were defined in [1] and [9] as generalizations of the class $\mathcal{UP}$. In the case of *Few*$\mathcal{P}$ the promise $R(N,x)$ states that on input x there are at most $p_N(|x|)$ accepting paths. The acceptance criterion $Q(N,x)$ states that there is an accepting path of N on input x. In the case of *Few* there is attached to each NPTM N a polynomial time machine M_N with the same time bound as N attached as a shut of clock. Note that by fixing a default machine, we can consider to any NPTM an attached PTM. The promise is the same as for *Few*. $Q(N,x)$ holds if M_N accepts $\langle x,i\rangle$ where i is the number of accepting paths of N on input x.

Notice that in all these cases the defining pairs fulfill A1 and A2. The most difficult case to verify is probably that the defining pair (Q,R) for *Few* fulfills A1. However, a universal machine U is given by a machine that on input $\langle N,x,0^s\rangle$ simulates N on x for s steps, where M_U is a machine that on input $\langle\langle N,x,0^s\rangle,i\rangle$ simulates M_N on $\langle x,i\rangle$ for at most s steps.

For the case of $\mathcal{NP}\cap$ *Sparse* the situation is not as straightforward. We define $R(N,x)$ to be true if $N=0^{N'}$ for some NPTM N' with attached polynomial time-bound $p_{N'}$ such that N' accepts at most $p_{N'}(|x|)$ strings of length $|x|$. Define $Q(N,x)$ to be true if $N=0^{N'}$ for some NPTM N' that has at least one accepting path on input x. Clearly $\mathcal{C}_{Q,R}$ is the class of sparse sets in $\mathcal{NP}$. To obtain a universal machine $U=0^{U'}$ we use as tupling function $\langle 0^{N'},x,0^s\rangle=0^{t(N',s,|x|)-|x|}1x$ where $t:\mathbb{N}^3\to\mathbb{N}$ is some standard tupling function with the additional property that $t(n,s,l)\geq l,s$ for all $n,s,l\geq 0$ (using the standard pairing function $p(i,j)=\binom{i+j}{2}+j$ one may define $t(n,s,l)=p(p(n,s),l)$). Now, on input w, U' verifies that $w=0^m1x$ and that there are N', s such that $m+|x|=t(N',s,|x|)$ and $s\geq p_{N'}(|x|)$. If this is not the case, then U' rejects. Otherwise U' simulates N' on x for at most $p_{N'}(|x|)$ steps. One can construct U' to be linearly time-bounded, so let the time bound $2n+1$ be encoded in its description. To see that A1 holds it remains to verify that $R(N,x)$ implies $R(U,\langle 0^{N'},x,0^s\rangle)$, i.e. that U' accepts at most $2l+1$ strings of length $l=|\langle 0^{N'},x,0^s\rangle|=t(N',s,|x|)+1$ if N' accepts at most $p_{N'}(|x|)$ strings of length $|x|$. Now as t is injective all strings of length l accepted by U' are of the form $0^{t(N',s,m)-|x'|}1x'$ where N',s,m are fixed for fixed l, and $|x'|=m$ where x' is accepted by N' in at most $p_{N'}(m)\leq s$ steps. So we are finished by observing that there are at most $p_{N'}(m)\leq s\leq l$ different x' of length m that are accepted by N'.

Lemma 24

(*i*) *Few, Few*$\mathcal{P}$*, and* $\mathcal{NP}\cap$ *Sparse have test sets in co-*$\mathcal{NP}$*.*

(*ii*) *For every* $k\geq 1$*,* $\Sigma_k^p\cap\Pi_k^p$ *has a test set which is* $\leq_m^p$*-reducible to* $\mathrm{SAT}_k\times\mathrm{TAUT}_k$*.*

Proof: All the test sets considered here are of the generic form

$$T_{(Q,R)}=\{\langle N,x,0^s\rangle \mid R(N,x) \text{ and } s\geq p_N(|x|)\}.$$

For *Few* and $Few\mathcal{P}$ one has to verify on input $\langle N, x, 0^s\rangle$ that N has at most $p_N(|x|)$ accepting paths on input x. But this can be easily done in $co\text{-}\mathcal{NP}$. For $\mathcal{NP} \cap Sparse$ one has to verify on input $\langle N, x, 0^s\rangle$ that there are at most $p_N(|x|)$ strings y of length $|x|$ such that N has an accepting path on y. Again, this can be easily decided in $co\text{-}\mathcal{NP}$.

In the second result, for the case $k = 1$ observe that for $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ the predicate $R(N, x)$ holds if there exists an accepting path α of N on input x, and if there is no pair β_1, β_2 of accepting paths of N on input x such that in the first nondeterministic step β_1 branches left and β_2 branches right. This shows that this test set is reducible to SAT $\times$ TAUT. The result for $k \geq 2$ is obtained in an analogous way. ∎

We now observe that $\mathcal{NP}$-assertions are useful for $\mathcal{NP} \cap Sparse$, and for $\Sigma_k^p \cap \Pi_k^p$, $k \geq 2$ (considering the machine models defined above). Hence, in order to get a many-one complete set for $\mathcal{NP} \cap Sparse$, and for $\Sigma_k^p \cap \Pi_k^p$, it suffices to find a test set with an optimal proof system.

Proposition 25 *$\mathcal{NP}$-assertions are useful for $\mathcal{NP} \cap Sparse$, and for $\Sigma_k^p \cap \Pi_k^p$, $k \geq 2$.*

Proof: For $\mathcal{NP} \cap Sparse$ observe that if N obeys the promise R on any $x \in B$ for some set $B \in \mathcal{NP}$ then setting $C = L_Q(N) \cap B$ yields $C \in \mathcal{NP} \cap Sparse$.

Basically the same argument holds for $\Sigma_k^p \cap \Pi_k^p$. Let N obey the $\Sigma_k^p \cap \Pi_k^p$-promise on $B \in \mathcal{NP}$. Then N consists of two Σ_k^p-machines N_1 and N_2 that are reached in the first nondeterministic branch. Let $L_i \in \Sigma_k^p$ denote the set accepted by N_i in a Σ_k^p-way (note that $L_Q(N) = L_1$). As N obeys the promise on B it follows that $B \setminus L_1 = L_2 \cap B$. Now let $C = L_1 \cap B$ (and hence, $C \cap B = L_Q(N) \cap B$). Clearly, $C \in \Sigma_k^p$, and further also $\overline{C} = L_2 \cup \overline{B} \in \Sigma_k^p$. This shows $C \in \Sigma_k^p \cap \Pi_k^p$. ∎

We can use Theorems 16 and 19 to get the following implications.

Corollary 26

- *If* TAUT *has a p-optimal proof system then Few and $Few\mathcal{P}$ have many-one complete sets.*
- *If* TAUT *has an optimal proof system then $\mathcal{NP} \cap Sparse$ has a many-one complete set.*
- *If* TAUT *and* SAT *have p-optimal proof systems, then $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ has a many-one complete set.*
- *For $k \geq 2$, if* TAUT_k *and* SAT_k *have optimal proof systems, then $\Sigma_k^p \cap \Pi_k^p$ has a many-one complete set.*

Using the above test sets one generically obtains length dependent test sets for *Few* and for *Few*$\mathcal{P}$ in *co*-$\mathcal{NP}$. Also for $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ one obtains a length dependent test set in Π_2^p. Hence, by applying Theorem 22 we get the following corollary.

Corollary 27

- *If* TAUT *has an optimal proof system then Few and Few$\mathcal{P}$ have complete sets under nonuniform many-one reducibility.*
- *If* TAUT_2 *has an optimal proof systems, then $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ has a complete set under nonuniform many-one reducibility.*

Test sets for probabilistic classes

We show now that the probabilistic complexity classes $\mathcal{BPP}$, $\mathcal{RP}$, and $\mathcal{ZPP}$ have test sets in Σ_2^p as well as in Π_2^p. We start by describing defining pairs (Q, R) for these promise classes that satisfy A1 and A2. Recall that for any NPTM N, p_N is the polynomial time bound associated with N. Let $\mathrm{Acc}(N, l, x)$ (resp., $\mathrm{Rej}(N, l, x)$) be the set of all paths $r \in \{0,1\}^l$ on which $N(x)$ accepts (rejects, respectively) after at most l steps. Notice that the two sets $\mathrm{Acc}(N, x) = \mathrm{Acc}(N, p_N(|x|), x)$ and $\mathrm{Rej}(N, x) = \mathrm{Rej}(N, p_N(|x|), x)$ form a partition of $\{0,1\}^{p_N(|x|)}$.

- For the case of $\mathcal{BPP}$, define $R(N, x)$ to be true if N is a NPTM such that $\|\mathrm{Acc}(N, x)\| \geq 2^{p_N(|x|)} \cdot 2/3$ or $\|\mathrm{Acc}(N, x)\| \leq 2^{p_N(|x|)}/3$ and let $Q(N, x)$ be true if $\|\mathrm{Acc}(N, x)\| > 2^{p_N(|x|)}/3$.
- For the case of $\mathcal{RP}$, define $R(N, x)$ to be true if N is a NPTM such that $\|\mathrm{Acc}(N, x)\| \geq 2^{p_N(|x|)}/2$ or $\|\mathrm{Acc}(N, x)\| = 0$ and let $Q(N, x)$ be true if $\|\mathrm{Acc}(N, x)\| > 0$.
- Since $\mathcal{ZPP} = \mathcal{RP} \cap co\text{-}\mathcal{RP}$, a machine model for $\mathcal{ZPP}$ can be obtained by combining two $\mathcal{RP}$-machines N_1 and N_2 which accept complementary languages into a machine N that in the first (nondeterministic) step, branches left to N_1 and right to N_2. So, the promise $R(N, x)$ states that on input x, N_1 and N_2 behave like an $\mathcal{RP}$-machine and that either N_1 or N_2 accepts but not both. $Q(N, x)$ holds if N_1 has an accepting path on input x.

Next we recall the definitions and basic properties of hashing that we need. Sipser [23] used universal hashing, originally invented by Carter and Wegman [10], to estimate (probabilistically) the size of a finite set X of strings.

A linear hash function h from Σ^m to Σ^k is given by a Boolean (k, m)-matrix (a_{ij}) and maps any string $x = x_1 \dots x_m$ to a string $y = y_1 \dots y_k$, where y_i is the inner product $a_i \cdot x = \sum_{j=1}^{m} a_{ij} x_j \pmod 2$ of the i-th row a_i and x.

Let $X \subseteq \Sigma^m$ and let h be a linear hash function from Σ^m to Σ^k. Then we say that h *hashes* X if for all pairs of different strings $x, y \in X$, $h(x) \neq h(y)$. More generally, if H is a family $(h_1, \ldots, h_s)$ of linear hash functions from Σ^m to Σ^k, then we say that H *hashes* X if for every $x \in X$ there is some i, $1 \leq i \leq s$, such that $h_i(x) \neq h_i(y)$, for all $y \in X - \{x\}$.

Note that the predicate "H hashes X" can be decided in *co-*$\mathcal{NP}$ provided that membership in X can be tested in $\mathcal{P}$. We denote the set of all families $H = (h_1, \ldots, h_k)$ of k linear hash functions from Σ^m to Σ^k by $\mathcal{H}(k, m)$.

As observed by Sipser, the size of a set $X \subseteq \Sigma^m$ can be estimated by checking for which values of k, X is hashable by some hash family $H \in \mathcal{H}(k, m)$.

Lemma 28 [23] *No hash family $H \in \mathcal{H}(k, m)$ can hash a set $X \subseteq \Sigma^m$ of cardinality $|X| > k2^k$. Furthermore, if $|X| \leq 2^k$, then some hash family $H \in \mathcal{H}(k, m)$ hashes X.*

The next two lemmas make use of Stockmeyer's refinement of the hashing technique [24]. Their proofs are straightforward (see, e.g., [15]).

Lemma 29 *Let $X \subseteq \{0,1\}^l$ and let $m = 1 + 72l$ and $k = 1 + m(l-2)$ be integers. Then the following implications hold.*

- *If there exists a hash family $H \in \mathcal{H}(k, lm)$ that hashes X^m, then $|X| \leq 2^l/3$.*
- *If $|X| \leq 2^l/4$, then some hash family $H \in \mathcal{H}(k, lm)$ hashes X^m.*

Lemma 30 *Let $X \subseteq \{0,1\}^l$ and let $m = 1 + 512l$ and $k = 1 + \lceil m(l + 1 - \log 3) \rceil$ be integers. Then the following implications hold.*

- *If there exists a hash family $H \in \mathcal{H}(k, lm)$ that hashes X^m, then $|X| \leq 2^l \cdot 3/4$.*
- *If $|X| \leq 2^l \cdot 2/3$, then some hash family $H \in \mathcal{H}(k, lm)$ hashes the set X^m.*

Proposition 31 *$\mathcal{BPP}$, $\mathcal{RP}$, and $\mathcal{ZPP}$ have test sets in Σ_2^p as well as in Π_2^p.*

Proof: For $\mathcal{BPP}$ we define the test set

$B = \{\langle N, x, 0^l \rangle \mid l = p_N(|x|)$ and for $m = 1 + 72l$ and $k = 1 + m(l-2)$ there is a hash family $H \in \mathcal{H}(k, lm)$ that hashes either $\mathrm{Acc}(N, l, x)$ or $\mathrm{Rej}(N, l, x)\,\}$.

Clearly, $B \in \Sigma_2^p$. Further, for any set $A \in \mathcal{BPP}$ there is an NPTM N such that for all inputs x and for $l = p_N(|x|)$ it holds that

$$\begin{aligned} x \in A &\Leftrightarrow \|\mathrm{Acc}(N, l, x)\| \geq 2^l \cdot 3/4, \\ x \notin A &\Leftrightarrow \|\mathrm{Rej}(N, l, x)\| \geq 2^l \cdot 3/4. \end{aligned}$$

Thus by Lemma 29 it follows that N passes test B. On the other hand, it also follows by Lemma 29 that if $\langle N, x, 0^l\rangle$ belongs to B then $l = p_N(|x|)$ and $R(N, x, 0^l)$ holds. This shows that B is a test set for $\mathcal{BPP}$.

Next we show that $\mathcal{BPP}$ has a test set in Π_2^p. In fact, consider the set

$$\begin{aligned} C = \{\langle N, x, 0^l\rangle \mid\ & l = p_N(|x|) \text{ and for } m = 1+512l \text{ and } k = 1+\lceil m(l+1-\log 3)\rceil \\ & \text{there is no hash family } H \in \mathcal{H}(k, lm) \text{ that hashes both } \mathrm{Acc}(N, l, x) \text{ and} \\ & \mathrm{Rej}(N, l, x)\,\} \end{aligned}$$

which belongs to Π_2^p. By Lemma 30 it is easy to see that C is a test set for $\mathcal{BPP}$. Moreover, it is not hard to adapt the above argument to get suitable test sets for $\mathcal{RP}$ and for $\mathcal{ZPP}$. ∎

As an immediate consequence of Proposition 31 and of Theorem 16 we obtain the following corollary.

Corollary 32 *If* SAT_2 *or* TAUT_2 *have a p-optimal proof system then* $\mathcal{BPP}$*,* $\mathcal{RP}$*, and* $\mathcal{ZPP}$ *have a many-one complete set.*

It is also not hard to show that $\mathcal{MA}$ has test sets Σ_2^p as well as in Π_2^p and that $\mathcal{AM}$ has test sets Σ_3^p as well as in Π_3^p. Since these classes can even use $\mathcal{NP}$-assertions, it follows that $\mathcal{MA}$ has a many-one complete set, if TAUT_2 or SAT_2 has an optimal proof system, whereas $\mathcal{AM}$ has a many-one complete set, if TAUT_3 or SAT_3 has an optimal proof system.

5 Completeness results for function classes

The results in Section 4 can be translated in a straightforward way to promise function classes. We just give a brief sketch. The definition of a promise R for function classes is the same as for languages, whereas the acceptance criterion Q is replaced by a function S mapping each pair (N, x) consisting of an NPTM N and a string x to the string $S(N, x)$. The function $F_S(N) : \Sigma^* \to \Sigma^*$ computed by N (when applying S) is given by

$$F_S(N)(x) = S(N, x).$$

R and S together define the function class

$$\mathcal{F}_{S,R} = \{F_S(N) \mid N \text{ is an } R\text{-machine}\}.$$

Conditions A1 and A2 translate to the corresponding conditions A1′ and A2′ for function classes. We just have to replace A1.1 and A2.1 by

A1′.1: $S(N, x) \;=\; S(U, \langle N, x, 0^s\rangle)$ and

A2′.1: $S(N, f_M(x)) \;=\; S(N \circ M, x)$

respectively. We use the following notion of many-one reducibility for functions: $g \le_m^p h$ if there is a function $f \in \mathcal{FP}$ such that $h(f(x)) = g(x)$ for any x in the domain of g. Notice that this notion is closely related to the notion of p-simulation (although g and h need not belong to $\mathcal{FP}$).

It is also straightforward to translate the definition of a test set. The notion of usefulness for a defining pair (S, R) for a function class $\mathcal{F}_{S,R}$ reads as follows. $\mathcal{A}$-assertions are called useful for (S, R) if for any language $B \in \mathcal{A}$ and any NPTM N the following holds: if $R(N, x)$ for any $x \in B$ then there is a function $f \in \mathcal{F}_{S,R}$ such that for all $x \in B$, $f(x) = S(N, x)$.

Theorems 16, 18, and 19 also translate to promise function classes. We first give the translation of the main equivalence of Theorem 16.

Theorem 33 *Let $\mathcal{F}$ be a promise function class and let (S, R) be a defining pair for $\mathcal{F}$ which fulfills A1$'$ and A2$'$. Then the following conditions are equivalent.*

1. *$\mathcal{F}$ has a (S, R)-test set with a p-optimal proof system.*

2. *$\mathcal{F}$ has a many-one complete set.*

Translating Theorem 19 to the functional setting yields the following sufficient condition for the existence of many-one complete functions.

Theorem 34 *Let $\mathcal{F}$ be a promise function class and let (S, R) be a defining pair for $\mathcal{F}$ which fulfills A1$'$. If $\mathcal{NP}$-assertions are useful for (S, R), then 1 implies 2:*

1. *$\mathcal{F}$ has a (S, R)-test set with an optimal proof system.*

2. *$\mathcal{F}$ has a many-one complete function.*

Razborov observes in [21] that the existence of an optimal proof system for TAUT would imply a the existence of a complete pair for the class of disjoint $\mathcal{NP}$-pairs. We recall that a pair (A, B) of $\mathcal{NP}$-languages belongs to this class when $A \cap B = \emptyset$. The reduction considered in [21] is a weak form of many-one reducibility. Formally, in [21] a pair (A, B) is said to many-one reduce to a pair (C, D) if for some $f \in \mathcal{FP}$, $f(A) \subseteq C$ and $f(B) \subseteq D$. By applying Theorem 33 we can improve the mentioned result showing that under assumption that TAUT has an optimal proof system, the class of disjoint $\mathcal{NP}$-pairs has a complete pair with respect to the following stronger notion of many-one reducibility: (A, B) strongly many-one reduces to (C, D) if for some $f \in \mathcal{FP}$, $f^{-1}(C) = A$ and $f^{-1}(D) = B$. We associate to each disjoint $\mathcal{NP}$-pair (A, B) a function $f_{(A,B)}$ as follows. For all $x \in \Sigma^*$,

$$f_{(A,B)}(x) = \begin{cases} 0 & x \in A, \\ 1 & x \in B, \\ \lambda & \text{otherwise.} \end{cases}$$

In a sense, the class of disjoint $\mathcal{NP}$-pairs corresponds to the function class of all these functions. This class can be defined as a promise class in the following way. An NPTM N is an R-machine if in the first nondeterministic step it branches to two $\mathcal{NP}$-machines N_1 and N_2 which accept disjoint languages. Therefore $R(N,x)$ holds if there is no pair α_1, α_2 of accepting paths of N on input x such that in the first nondeterministic step α_1 branches left and α_2 branches right. If there is an accepting path branching left, the value of $S(N,x)$ is 0. Otherwise, $S(N,x) = 1$ if there is an accepting path branching right, and $S(N,x) = \lambda$ if there isn't any accepting path. This defines the class $\mathcal{F}_{S,R}$ which has a $\leq_m^p$-complete function if and only if there is a strongly many-one complete disjoint $\mathcal{NP}$-pair. It is easy to see that $\mathcal{F}_{S,R}$ has a test set in *co-*$\mathcal{NP}$ and that $\mathcal{NP}$-assertions are useful for $\mathcal{F}_{S,R}$. Therefore, Theorem 34 gives us the following consequence.

Corollary 35 *If* TAUT *has an optimal proof system, then there is a pair that is strongly many-one complete for the class of all disjoint* $\mathcal{NP}$*-pairs.*

6 Sufficient conditions

In this section we investigate conditions which imply the existence of (p-)optimal proof systems. We first make an observation which allows us to infer the existence of a p-optimal proof system for a recursively enumerable set L, provided that there are complete functions for certain promise function classes. Secondly, we will see that collapses of tally sets at the double exponential-time level imply the existence of (p-)optimal proof systems for TAUT_k.

For any recursively enumerable set L, the function class $\mathcal{PS}_L = \{f \in \mathcal{FP} \mid f(\Sigma^*) \subseteq L\}$ is the $\leq_m^p$-closure of the class $\{f \in \mathcal{FP} \mid f(\Sigma^*) = L\}$ that consists of all proof systems for L. Clearly, $\mathcal{PS}_L$ has a $\leq_m^p$-complete function if and only if there is a p-optimal proof system for L. Furthermore the class $\mathcal{PS}_L$ is easily described as a promise function class by the following defining pair (S,R) which fulfills A1$'$ and A2$'$.

1. $R(N,x)$ holds if on input x, N only makes deterministic moves, and if N accepts then the string y written on its tape is in L.

2. $S(N,x) = y$ where y is the string produced by N on input x on its leftmost accepting nondeterministic computation.

It is also possible to describe this class using the tree structure approach for promise classes. Here, the idea is to allow only special trees (called combs in [6]) which represent polynomial-time computable functions.

An (S,R)-test set is given by the set

$$T_L = \{\langle N, x, 0^s\rangle \mid R(N,x) \text{ and } s \geq p_N(|x|)\}\,.$$

It is easy to see that $T_L \leq_m^p L$ for $L \notin \{\emptyset, \Sigma^*\}$ via a PTM M which on input $\langle N, x, 0^s\rangle$ simulates N for at most s steps as follows. If N does only perform deterministic moves, then M behaves as N and produces N's output if N accepts, if N rejects then M outputs a fixed string $y \in L$. Otherwise, M outputs a fixed string $y \notin L$. Notice that also $L \leq_m^p T_L$ via $f : x \mapsto \langle N_{\mathrm{id}}, x, 0^{|x|}\rangle$, where N_{id} is an NPTM computing the identity mapping; implying that $T_L \equiv_m^p L$.

Combining these observations with Theorem 33 we obtain the following theorem.

Theorem 36 *Let $L \subseteq \Sigma^*$. Then the following statements are equivalent.*

1. *There is a p-optimal proof system for L.*
2. *Every promise function class $\mathcal{F}$ which has a defining pair (S, R) fulfilling A1′ and A2′, and further possesses an (S, R)-test set which is $\leq_m^p$-reducible to L has a $\leq_m^p$-complete function.*
3. *$\mathcal{PS}_L$ has a $\leq_m^p$-complete function.*

It would be interesting to know whether a similar theorem holds when considering just language classes instead of function classes.

We now give a sufficient condition for the existence of a (p-)optimal proof system for TAUT_k. Observe that $\mathcal{PS}_{\mathrm{TAUT}}$ has a length dependent test set in *co-*$\mathcal{NP}$. Hence (using Lemma 9 and Theorem 34 (resp. 33)), there is a (p-)optimal proof system for TAUT provided that any tally set in *co-*$\mathcal{NP}$ is already in $\mathcal{NP}$ ($\mathcal{P}$). Together with observations from [5] that relate sets in $\mathcal{E}$, $\mathcal{NE}$ to tally sets in $\mathcal{P}$, $\mathcal{NP}$ this gives a proof of Theorem 6. Actually the idea to this proof of Theorem 6 dates back to [20] where 'finitistic consistency statements' roughly correspond to elements of a length dependent test set for $\mathcal{PS}_{\mathrm{TAUT}}$. We can weaken the needed assumption if we consider (intuitively) super-tally sets instead of just tally sets, where we call a set T *super-tally* (in symbols: $T \in$ *Super-Tally*) if T is a subset of $\{0^{2^{2^n}} \mid n \geq 0\}$.

Theorem 37 *If any super-tally set in $\mathcal{NP}$ is already in $\mathcal{P}$, then* TAUT *has a p-optimal proof system.*

Proof: We assume some standard enumeration M_1, M_2, M_3, ... of (encodings of) deterministic Turing transducers with binary input alphabet such that for a given triple $\langle M_i, x, 0^k\rangle$, up to k steps of the computation of M_i on input x can be efficiently simulated. Let $i(k)$ denote the largest exponent i such that 2^i divides k and consider the language

$$T = \{\, 0^{2^{2^k}} \mid \text{on any input, } M_{i(k)} \text{ either stops after at most } 2^{2^k} \text{ steps and outputs some tautology or } M_{i(k)} \text{ runs for more than } 2^{2^k} \text{ steps}\,\}.$$

In order to decide whether a given string $0^n = 0^{2^{2^k}}$ belongs to T, it suffices to simulate $M_{i(k)}$ on any input of length at most $n+1$ for at most $n+1$ steps. This shows that $\overline{T} \cap \{ 0^{2^{2^k}} \mid k \geq 0 \} \in \mathcal{NP}$ and thus, by the assumption that any super-tally set in $\mathcal{NP}$ is already in $\mathcal{P}$, T can be decided in $\mathcal{P}$. We claim that the following transducer computes a p-optimal proof system h for TAUT.

> **input** $\langle 0^n, w \rangle$
> **if** $0^n \in T$ **then**
>   determine k such that $n = 2^{2^k}$
>   **if** $M_{i(k)}$ stops on input w in at most n steps **then**
>   output $M_{i(k)}(w)$ and halt;
> (otherwise reject).

Since, as is not hard to see, $h(\Sigma^*) \subseteq$ TAUT and $h \in \mathcal{FP}$, it only remains to show that h is p-optimal. Let g be any proof system for TAUT, computed by some deterministic Turing transducer M_i in time bounded by some polynomial p. Then any g-proof w can be translated into an h-proof by the mapping $w \mapsto \langle 0^{2^{2^k}}, w \rangle$, where k is the smallest integer $k_j = (2j+1)2^i$, $j \geq 0$, such that $p(|w|) \leq 2^{2^{k_j}}$. Since

$$2^{2^{k_{j+1}}} = \left(2^{2^{k_j}}\right)^c,$$

where $c = 2^{2 \cdot 2^i}$, it follows that $2^{2^k} < p(|w|)^c$, implying that the translation $w \mapsto \langle 0^{2^{2^k}}, w \rangle$ is computable in polynomial time. ∎

It is interesting to note that the above proof still goes through if we define a set T to be super-tally if it is a subset of $\{0^{c^{c^k}} \mid k \geq 0\}$ where $c \geq 2$ is an arbitrary integer constant. However, in our proof, we cannot allow T to be any sparser. To see why, let us just try to replace the function $j \mapsto 2^{2^{(2j+1)2^i}}$ by some other function $f(j)$ (where f as well as the constant c below might depend on i). This would guarantee that the new set T is a subset of $\{0^{f(j)} \mid j \geq 0\}$. On the other hand, a necessary condition for the proof to work is that for some constant c, $f(j+1) \leq f(j)^c$, implying that $f(j) \leq f(0)^{c^j}$.

A similar proof shows that there is an optimal proof system for TAUT, provided that any super-tally set in $\mathcal{NP}$ is also in *co*-$\mathcal{NP}$.

Let $\mathcal{EE} = \mathrm{DTIME}(2^{O(2^n)})$ (cf. [12]), $\mathcal{EEE} = \mathrm{DTIME}(2^{O(2^{2^n})})$ and let $\mathcal{NEE}$, $\mathcal{NEEE}$ be their nondeterministic counterparts. Using a technique in the style of [5] it is easy to see that each tally language in $\mathcal{EE}$ ($\mathcal{NEE}$) translates to a super-tally language in $\mathcal{P}$ (respectively $\mathcal{NP}$) and vice versa. Thus a collapse of tally sets at the $\mathcal{EE}$-level (as, e.g., $\mathcal{NEE} \cap$ *Tally* $\subseteq \mathcal{EE}$ or $\mathcal{NEE} \cap$ *Tally* $\subseteq$ *co*-$\mathcal{NEE}$) corresponds to a collapse for super-tally sets at the $\mathcal{P}$-level (i.e., $\mathcal{NP} \cap$ *Super-Tally* $\subseteq \mathcal{P}$ and $\mathcal{NP} \cap$ *Super-Tally* $\subseteq$ *co*-$\mathcal{NP}$, respectively). As a consequence we can state the following corollary.

Corollary 38

If $\mathcal{NEE} \cap \text{Tally} \subseteq \mathcal{EE}$ *then* TAUT *has a p-optimal proof system.*

If $\mathcal{NEE} \cap \text{Tally} \subseteq$ *co-*$\mathcal{NEE}$ *then* TAUT *has an optimal proof system.*

Surprisingly, it seems hard to improve the above corollary further to the triple exponential time level. In fact, it is stated in [4] that there is a relativized world in which $\mathcal{NEEE} = \mathcal{EEE}$ but TAUT does not have an optimal proof system.

Finally, a generalization of Theorem 37 to any level of the polynomial time hierarchy yields the following sufficient conditions for the existence of a (p-)optimal proof system for TAUT_k.

Theorem 39

If any super-tally set in Π_k^p *is in* $\mathcal{P}$ *then* TAUT_k *has a p-optimal proof system.*

If any super-tally set in Π_k^p *is in* $\mathcal{NP}$ *then* TAUT_k *has an optimal proof system.*

References

[1] E. Allender. *Invertible functions.* PhD thesis , Georgia Institute of Technology, 1985.

[2] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science #11. Springer-Verlag, 1988.

[3] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs on Theoretical Computer Science #22. Springer-Verlag, 1990.

[4] S. Ben-David and A. Gringauze. On the existence of optimal proof systems and oracle-relativized propositional logic. Technical Report TR98-021, Electronic Colloquium on Computational Complexity, 1998.

[5] R. V. Book. Tally languages and complexity classes. *Information and Control* **26**:186–193, 1974.

[6] B. Borchert. *Predicate Classes, Promise Classes, and the Acceptance Power of Regular Languages.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1994.

[7] D. P. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, **104**:263–283, 1992.

[8] S. Buss. Lectures on Proof Theory. Technical Report No. SOCS-96.1, McGill University, 1996. (http://www.cs.mcgill.ca/~denis/TR.96.1.ps.gz)

[9] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory* **23**:95–106, 1990.

[10] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, **18**:143–154, 1979.

[11] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic* **44**:36–50, 1979.

[12] J. Hartmanis, N. Immerman and V. Sewelson. Sparse sets in $\mathcal{NP} - \mathcal{P}$: EXPTIME versus NEXPTIME. *Information and Control* **65**:158–181, 1985.

[13] J. Hartmanis and L. Hemachandra Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, **58**:129–142, 1988.

[14] J. Köbler and J. Messner. Complete Problems for Promise Classes by Optimal Proof Systems for Test Sets In *Proc. 13th Annual IEEE Conference on Computational Complexity, CC 98*, 132–140, 1998.

[15] J. Köbler. *Lowness-Eigenschaften und Erlernbarkeit von Booleschen Schaltkreisklassen.* Habilitation Thesis, Universität Ulm, 1995.

[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*, Lecture Notes in Computer Science #176, 364-369, Springer-Verlag, 1984.

[17] J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.

[18] J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic* **54**: 1063–1079, 1989.

[19] J. Messner and J. Torán. Optimal proof systems for propositional logic and complete sets. In *Proc. 15th Symposium on Theoretical Aspects of Computer Science '98*, Lecture Notes in Computer Science #1373, 477–487. Springer-Verlag, 1998.

[20] P. Pudlák. On the length of proofs of finitistic consistency statements in first order theories. *Logic Colloquium'84* (J. B. Paris et al., editors), North-Holland, Amsterdam, pp. 165–196, 1986

[21] A. A. Razborov. On provably disjoint NP-pairs. Technical Report RS-94-36, Basic Research in Computer Science Center, Aarhus, 1994.

[22] 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*, Lecture Notes in Computer Science #1279, 423–428. Springer-Verlag, 1997.

[23] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th ACM Symposium on Theory of Computing*, pp. 330–335. ACM Press, 1983.

[24] L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, **14**(4):849–861, 1985.

[25] A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic* **1**:425-467, 1995.

[26] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, **5**:20–23, 1976.

[27] N. K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestija Rossijskoj Akademii Nauk*, **57**:51–90, 1993. In Russian.