Article:
Beyersdorff, O, Koebler, J and Messner, J (2009) Nondeterministic functions and the existence of optimal proof systems. Theoretical Computer Science, 410 (38-40). 3839 - 3855 . ISSN 0304-3975

https://doi.org/10.1016/j.tcs.2009.05.021

Nondeterministic Functions and the Existence of Optimal Proof Systems*

Olaf Beyersdorff[1], Johannes Köbler[2], and Jochen Messner[3]

[1] Institut für Theoretische Informatik, Leibniz-Universität Hannover, Germany
beyersdorff@thi.uni-hannover.de
[2] Institut für Informatik, Humboldt-Universität zu Berlin, Germany
koebler@informatik.hu-berlin.de
[3] Fakultät für Elektronik und Informatik, Hochschule Aalen, Germany
jochen.messner@htw-aalen.de

Abstract. We provide new characterizations of two previously studied questions on nondeterministic function classes:

Q1: Do nondeterministic functions admit efficient deterministic refinements?
Q2: Do nondeterministic function classes contain complete functions?

We show that Q1 for the class NPMV_t is equivalent to the question whether the standard proof system for SAT is p-optimal, and to the assumption that every optimal proof system is p-optimal. Assuming only the existence of a p-optimal proof system for SAT, we show that every set with an optimal proof system has a p-optimal proof system. Under the latter assumption, we also obtain a positive answer to Q2 for the class NPMV_t.
An alternative view on nondeterministic functions is provided by disjoint sets and tuples. We pursue this approach for disjoint NP-pairs and its generalizations to tuples of sets from NP and coNP with disjointness conditions of varying strength. In this way, we obtain new characterizations of Q2 for the class NPSV. Question Q1 for NPSV is equivalent to the question whether every disjoint NP-pair is easy to separate. In addition, we characterize this problem by the question whether every propositional proof system has the effective interpolation property. Again, these interpolation properties are intimately connected to disjoint NP-pairs, and we show how different interpolation properties can be modeled by NP-pairs associated with the underlying proof system.

1 Introduction

Most computational tasks are naturally formulated as functional problems, i.e., for a given input a solution to the problem instance has to be computed. Quite in contrast, computational complexity theory mainly studies language problems and their associated complexity classes. Of course, by studying the *undergraph* $\{\langle x, y\rangle \mid y \leq f(x)\}$ of a function f, every functional problem can be transformed into a corresponding decision version, which justifies the focus on language complexity. On the other hand, some computational phenomena are most naturally addressed in the functional setting, and this particularly applies to nondeterministic functions (cf. [54] for a beautiful argument on this theme).

Prominent questions in the functional context are in particular:

* Part of the results of this paper appeared in an extended abstract in the proceedings of the conference FSTTCS 2000 [27]. This work was done while the first author was at Humboldt University Berlin and the third author was at Ulm University. Research was partially supported by DFG grant KO 1053/5-1.

Q1: Do nondeterministic functions possess efficient deterministic refinements?
Q2: Do nondeterministic function classes possess complete functions?

During the last decade these problems have been intensively studied for a variety of function classes (cf. [53] for a comprehensive taxonomy or [23, 29] for equivalent characterizations). Question Q1 is important in connection with cryptographic applications, as Q1 (for the function class NPMV_t) is equivalent to the question whether all polynomial-time computable onto honest functions are invertible in polynomial time. The question was further characterized by Fenner, Fortnow, Naik, and Rogers [14] by a number of previously studied complexity-theoretic assumptions, and they named the list of these equivalences as "Q" (cf. also [16]). Determining the precise strength of Q seems to be intricate. On the one hand, Q has unlikely collapse consequences such as $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$. On the other hand, Q does not seem as strong as to imply a collapse of the polynomial hierarchy [10].

In this paper we will argue that the above two questions on function classes are closely connected to disjoint NP-pairs and their generalizations, as well as to problems about proof systems. Disjoint NP-pairs have recently been intensively studied [42, 17–21, 49, 5, 3, 4], mainly, because they are suitable objects to model the security of cryptosystems [22, 33], and further, because they are intimately connected to propositional proof systems [45, 42, 19, 21, 3].

Nondeterministic functions were already linked to disjoint coNP-pairs by Fenner et al. [14]. Here we will extend this connection to further function classes and disjoint NP-pairs as well as tuples of disjoint NP-sets (cf. [5]) and disjoint coNP-pairs. This correspondence provides an alternative view on disjoint NP-pairs and allows elegant characterizations of Questions Q1 and Q2 above. Namely, Q1 is equivalent to the statement that every disjoint NP-pair is easy to separate, while Q2 is equivalent to the problem, whether the class of disjoint NP-pairs (and its generalizations) possess complete elements. In the context of NP-pairs, this question was posed by Razborov [45], and it has been intensively studied during the last years [17, 18, 5, 3, 4]. Our characterizations restate and unify some of these recent results in terms of nondeterministic functions.

Another important connection of nondeterministic functions (and equivalently of disjoint sets) is to the field of proof systems, as introduced for arbitrary languages by Cook and Reckhow [11]. We will show that in this setting, Question Q1 (for functions from NPSV) can be restated as the question, whether all propositional proof systems satisfy the effective interpolation property (cf. [31, 33]). This again is equivalent to the statement that every disjoint NP-pair is P/poly-separable, which in turn implies that $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$ and $\mathsf{UP} \subseteq \mathsf{P/poly}$.

Similarly, we also provide another characterization of Q (or equivalently, Question Q1 for functions from NPMV_t). Namely, we investigate the problem, whether the standard proof system *sat* for SAT is p-optimal[4], where proofs in *sat* are given by a satisfying assignment for the formula in question. We show that this question is equivalent to the assertion Q, and it is further characterized

[4] Pavel Pudlák posed this question during the discussion after Zenon Sadowski's talk at CSL'98 [47]

by the statement that the two common notions of reductions between proof systems, i.e., simulations [32] and p-simulations [11], coincide. Thus Q is also equivalent to the statement that every optimal proof system is p-optimal. Under the weaker assumption of the mere existence of a p-optimal proof system for SAT we can still show that every language with an optimal proof system also has a (possibly different) p-optimal proof system.

The (likely) assumption that there are no p-optimal proof systems for SAT (as well as for TAUT) also has some practical implications due to its connection to the existence of optimal algorithms (cf. [32, 48, 49, 36]). Note that usually a decision algorithm for SAT also provides a satisfying assignment for any positive instance. However, if *sat* is not p-optimal, then no decision algorithm for SAT that produces satisfying assignments for positive instances can be optimal (cf. Theorem 16). In fact, a stronger consequence can be derived: if *sat* is not p-optimal, then there is a non-sparse set of easy instances from SAT for which it is hard to produce a satisfying assignment (cf. Theorem 20).

It has been observed in [46, 28] that (p-)optimal proof systems for certain languages can be used to define complete sets for certain promise classes. For example, if TAUT has an optimal proof system, then $\mathsf{NP} \cap \mathit{Sparse}$ has a many-one complete set, and if TAUT as well as SAT have a p-optimal proof system, then $\mathsf{NP} \cap \mathsf{coNP}$ has a complete set. We complete this picture here by showing that already a p-optimal proof system for SAT can be used to derive completeness consequences.

In particular, we prove that a p-optimal proof system for SAT implies complete functions for NPMV_t (which in turn implies complete disjoint coNP-pairs). Further, the existence of an optimal proof system for TAUT implies the existence of complete functions for NPkV (or equivalently, complete tuples of NP-sets with some disjointness conditions). And finally, the existence of optimal proof systems for TAUT and p-optimal proof systems for SAT implies the existence of complete functions for NPSV_t (or equivalently, complete sets for $\mathsf{NP} \cap \mathsf{coNP}$).

Overview of the Paper

This paper is organized as follows. After fixing notation and reviewing relevant definitions about function classes, proof systems, and disjoint tuples (Sect. 2), we start in Sect. 3 by exploring the connections between nondeterministic functions and pairs (as well as tuples) of disjoint sets. Particular attention is directed towards the problem of the existence of complete functions and pairs for the respective classes (Question Q2 above).

Section 4 is devoted to Question Q1 above, i.e., whether functions from NPSV possess total refinements in FP or $\mathsf{FP/poly}$. It turns out that this questions is intimately connected to different interpolation properties of propositional proof systems, and we characterize these interpolation properties by disjoint NP-pairs, associated with the proof system.

In Sect. 5 we investigate whether the standard proof system *sat* for SAT is p-optimal. We show this question to be equivalent to the assertion Q from [14] (and hence to Question Q1 for NPMV_t). In addition we provide several

new characterizations of this problem in terms of simulations and optimal algorithms.

Finally, in Sect. 6 we analyse the weaker question whether there exists a p-optimal proof system for SAT. We show that this is equivalent to the statement that every language with an optimal proof system also has a p-optimal proof system, and derive some collapse consequences from these assumptions.

2 Preliminaries and Notation

Let $\Sigma = \{0,1\}$. We denote the cardinality of a set A by $\|A\|$ and the length of a string $x \in \Sigma^*$ by $|x|$. The empty word is denoted by λ. FP is the class of (partial) functions that can be computed in polynomial time. A set S is called *sparse* if the cardinality of $S \cap \Sigma^n$ is bounded above by a polynomial in n. S is called *printable* if there exists a function in FP which on input 1^n outputs all elements in S of length n. We use $\langle \cdot, \ldots, \cdot \rangle$ to denote a standard polynomial-time computable tupling function. For the definitions of standard complexity classes like P, NP etc. we refer to the monographs [2] and [39].

A function h is called FP-*invertible* if there is a function $f \in \mathsf{FP}$ that *inverts* h, i.e., $h(f(y)) = y$ for each y in the range of h. A function h is *honest* if for some polynomial p, $p(|h(x)|) \geq |x|$ holds for all x in the domain of h. We call a function g an *extension* of a function f if $f(x) = g(x)$ for any x in the domain of f. A function $r : \mathbb{N} \to \mathbb{N}$ is called *super-polynomial* if for each polynomial p, $r(n) > p(n)$ for almost every $n \geq 0$. A set $B \in \mathsf{P}$ with $B \subseteq L$ is called a P-*subset of* L.

2.1 Nondeterministic Function Classes

A nondeterministic polynomial-time Turing machine (NPTM, for short) is a Turing machine N such that for some polynomial p, every accepting path of N on any input of length n is at most of length $p(n)$. A nondeterministic transducer is a nondeterministic Turing machine T with a write-only output tape. On input x, T outputs $y \in \Sigma^*$ (in symbols: $T(x) \mapsto y$) if there is an accepting path on input x along which y is written on the output tape. Hence, the function f computed by T on Σ^* could be multi-valued and partial. Using the notation of [9, 53] we denote the set $\{y \mid f(x) \mapsto y\}$ of all output values of T on input x by set-$f(x)$.

The class of all multi-valued, partial functions computable by some nondeterministic polynomial-time transducer T is denoted by NPMV. But also various subclasses of NPMV are of interest. NPSV is the class of functions f in NPMV that are *single-valued*, i.e., $\|\text{set-}f(x)\| \leq 1$. Thus, the functions from NPSV are functions in the usual sense, and we use $f(x)$ to denote the unique string in set-$f(x)$. Relaxing the condition $\|\text{set-}f(x)\| \leq 1$ by allowing $\|\text{set-}f(x)\| \leq k$ for some fixed number $k \geq 1$ leads to the classes NPkV, defined in [38, 14]. Even more generally, Fenner, Fortnow, Naik, and Rogers [14] considered functions f where the cardinality of set-$f(x)$ is bounded by a function $g(x)$ rather than a constant. For a function g, this function class is denoted by NPgV.

The *domain* of a multi-valued function is the set of those inputs x where set-$f(x) \neq \emptyset$. A function is called *total* if its domain is Σ^*. For a function class $\mathcal{F}$ we denote by $\mathcal{F}_t$ the class of total functions in $\mathcal{F}$. We use $\mathcal{F}_t \subseteq_c \mathsf{FP}$ to indicate that for any $g \in \mathcal{F}_t$ there is a total function $f \in \mathsf{FP}$ that is a *refinement* of g, i.e., $f(x) \in$ set-$g(x)$ for all $x \in \Sigma^*$. Occasionally it is useful to explicitly indicate the *range* of a multi-valued function in the notation. To do this we collect in the class $\mathcal{F}_A$ all functions from $\mathcal{F}$ which range over subsets of $A \subseteq \Sigma^*$,i.e., set-$f(x) \subseteq A$ for all $x \in \Sigma^*$.

Reductions for nondeterministic functions can be considered from a whole spectrum of reductions, ranging from many-one to Turing reductions. We start with a rather strong notion of many-one reducibility:

Definition 1. *A multi-valued function h* many-one reduces *to a multi-valued function g (denoted by $h \leq^p_m g$), if there is a function $f \in \mathsf{FP}$ such that for every $x \in \Sigma^*$ set-$g(f(x)) =$ set-$h(x)$.*

On the other side of the spectrum we use Turing reductions of which different versions are considered in the literature (cf. [15, 50, 53]). An oracle Turing transducer M may access a single-valued function oracle g by repeatedly querying function values. On such a query x, the oracle returns the unique value from set-$g(x)$ if x is in the domain of g, otherwise M stops without any output. Using this machine model, we define Turing reductions:

Definition 2. *A multi-valued function h* Turing reduces *to a multi-valued function g, if there is a deterministic oracle transducer M such that for each single-valued refinement g' of g, $M^{g'}$ computes a single-valued refinement of h.*

In between these two kinds of reducibilities, it is natural to consider other variants of many-one reductions, for example, by allowing post-computations (cf. [34, 56]). As we will formulate most of our results in the strongest possible way (by using Turing reductions in hypotheses and many-one reductions in conclusions), they also apply to intermediate reducibilities.

2.2 Proof Systems

Cook and Reckhow [11] defined the notion of an abstract *proof system* for a set $L \subseteq \Sigma^*$ as a (possibly partial) polynomial-time computable function $h : \Sigma^* \to \Sigma^*$ with range L. In this setting, an *h-proof* for the membership of φ to L is given by a string w with $h(w) = \varphi$. We use the notation $h \vdash_{\leq m} \varphi$ to indicate that there exists an h-proof of φ of size $\leq m$. Proof systems for the set of all tautologies TAUT are called *propositional proof systems*.

To compare the relative strength of different proof systems, Cook and Reckhow [11] introduced the notion of p-simulation. A proof system h *p-simulates* a proof system g if g-proofs can be translated into h-proofs in polynomial time, i.e., there is a polynomial-time computable function f such that for each v in the domain of g, $h(f(v)) = g(v)$. Similarly, h is said to *simulate* g if for each g-proof v there is an h-proof w of length polynomial in the length of v with $h(w) = g(v)$. A proof system for a set L is called *(p-)optimal* if it (p-)simulates every proof system for L (cf. [32]). It is a natural question to ask whether a set

L has a p-optimal (or at least an optimal) proof system. Note that a p-optimal proof system has the advantage that from any proof in another proof system one can efficiently obtain a proof for the same instance in the p-optimal proof system. Hence, any method that is used to compute proofs in some proof system can be reformulated to yield proofs in the p-optimal proof system with little overhead.

2.3 Disjoint Pairs and Tuples

For a class $\mathcal{C}$ of sets we call a tuple $(A_1, \ldots, A_l)$ of sets $A_1, \ldots, A_l \in \mathcal{C}$ a *disjoint $\mathcal{C}$-tuple* if $A_i \cap A_j = \emptyset$ for all $1 \leq i < j \leq l$. For $l = 2$ we just say that (A_1, A_2) is a *disjoint $\mathcal{C}$-pair*, or simply a $\mathcal{C}$-pair. For such a disjoint pair (A_1, A_2) of languages let us say that (A_1, A_2) is *$\mathcal{D}$-separable* if there is a language $S \in \mathcal{D}$ which *separates* (A_1, A_2), i.e., $A_1 \subseteq S$ and $A_2 \cap S = \emptyset$ (cf. [22]).

Grollmann and Selman [22] introduced a notion of many-one reducibility between disjoint NP-pairs, a stronger version of which was studied in [28]. In [5] these reductions were generalized to tuples as follows. Let $(B_1, \ldots, B_l)$ and $(C_1, \ldots, C_l)$ be disjoint NP-tuples. The tuple $(B_1, \ldots, B_l)$ many-one reduces to the tuple $(C_1, \ldots, C_l)$ if there is a function $f \in \mathsf{FP}$ such that $f(B_i) \subseteq C_i$ for $i = 1, \ldots, l$. If, in addition, f also respects the complement of the union $B_1 \cup \cdots \cup B_l$, i.e., $f(\overline{B_1 \cup \cdots \cup B_l}) \subseteq \overline{C_1 \cup \cdots \cup C_l}$, then we call the reduction *strong*. We denote these reductions by $\leq_p$ and $\leq_s$, respectively. We remark that f strongly reduces $(B_1, \ldots, B_l)$ to $(C_1, \ldots, C_l)$ if and only if f is a many-one polynomial-time reduction of B_i to C_i for each $i \in \{1, \ldots, l\}$.

3 Nondeterministic Function Classes and Tuples of NP-Sets

There is a direct correspondence between nondeterministic functions and tuples of NP-sets, which we will explore in this section. The simplest case is provided by functions from NPSV_t and languages from $\mathsf{NP} \cap \mathsf{coNP}$. With respect to this relation, Selman [53] and Hemaspaandra et al. [24] have shown that $\mathsf{NPSV}_t = \mathsf{FP}_t^{\mathsf{NP} \cap \mathsf{coNP}}$, from which Fenner et al. [14] concluded that $\mathsf{NPSV}_t \subseteq \mathsf{FP}$ holds if and only if $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$. We complete the picture by showing that this correspondence also extends to the question of the existence of complete problems.

Proposition 3. *The following conditions are equivalent:*

1. $\mathsf{NP} \cap \mathsf{coNP}$ *has a many-one complete set.*
2. NPSV_t *has a many-one complete function.*
3. NPSV_t *has a Turing complete function.*

Proof. For the implication $1 \Rightarrow 2$, assume that C is many-one complete for $\mathsf{NP} \cap \mathsf{coNP}$. Hence, $\mathsf{NPSV}_t = \mathsf{FP}_t^{\mathsf{NP} \cap \mathsf{coNP}} = \mathsf{FP}_t^C$. But FP_t^C has a many-one complete function for any C, and therefore also NPSV_t has a many-one complete function.

The implication $2 \Rightarrow 3$ is immediate. For $3 \Rightarrow 1$, let us assume that h is a Turing complete function for NPSV_t. Since $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}^{\mathsf{NPSV}_t}$ it follows that $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}^h$, and P^h has a many-one complete set for any function h. $\square$

Now let us consider the function class NPSV. In the same way as NPSV_t corresponds to the language class $\mathsf{NP} \cap \mathsf{coNP}$, the function class NPSV corresponds to the class of all disjoint NP-pairs. In fact, if we denote the class of all 0,1-valued functions in NPSV by $\mathsf{NPSV}_{\{0,1\}}$, then any function $h \in \mathsf{NPSV}$ can be identified with the disjoint NP-pair (D_0, D_1) where

$$D_b = \{x \in \Sigma^* \mid h(x) \mapsto b\}.$$

Generalizing this observation, for some finite set $A = \{a_1, \ldots, a_l\} \subset \Sigma^*$ containing $l \geq 2$ elements, the class NPSV_A of all functions in NPSV taking only values in A corresponds to the class of all disjoint l-tuples of NP-sets, studied in [5]. If f is a function from NPSV_A, then we can define a disjoint l-tuple of NP-sets $D^f = (D_1^f, \ldots, D_l^f)$ by $D_i^f = \{x \in \Sigma^* \mid f(x) \mapsto a_i\}$. Conversely, a disjoint l-tuple of NP-sets $(D_1, \ldots, D_l)$ defines a nondeterministic function as follows. Let M_i be nondeterministic polynomial-time machines that decide the sets D_i, respectively. The machine $M(x)$ first nondeterministically chooses an index $i \in \{1, \ldots, l\}$ and outputs the value a_i if the machine $M_i(x)$ accepts. Thus M computes a function f from the class NPSV_A such that $D^f = (D_1, \ldots, D_l)$.

This correspondence between functions from NPSV_A and disjoint tuples of NP-sets also extends to the respective simulations, namely:

Proposition 4. *Let A be a finite subset of Σ^*, and let f and g be functions from NPSV_A. Then $f \leq_m^p g$ if and only if $D^f \leq_s D^g$.*

Thus, for example, the class of disjoint NP-pairs has a strongly many-one complete pair if and only if $\mathsf{NPSV}_{\{0,1\}}$ has a many-one complete function. As shown in the next theorem, this is even equivalent to the assumption that NPSV has a many-one complete function. In addition, the theorem gives alternative and easier proofs for some results from [5] on disjoint NP-tuples.

Theorem 5. *The following statements are equivalent.*

1. NPSV *has a many-one complete function.*
2. *For all polynomial-time decidable sets $A \subseteq \Sigma^*$, the class NPSV_A has a many-one complete function.*
3. *For some set $A \subseteq \Sigma^*$ with at least two elements, the class NPSV_A has a many-one complete function.*
4. *For all numbers $l \geq 2$ there exist $\leq_s$-complete disjoint l-tuples of NP-sets.*
5. *For some number $l \geq 2$ there exist $\leq_s$-complete disjoint l-tuples of NP-sets.*
6. *There is a $\leq_s$-complete disjoint NP-pair.*

This list can be extended by the analogous versions of items 1 to 3 where many-one reducibility is replaced by Turing reducibility.

Proof. To obtain the above equivalences we will verify the following implications: $1 \Rightarrow 2 \Rightarrow 4 \Rightarrow 6 \Rightarrow 5 \Rightarrow 3 \Rightarrow 1$, of which the implications $4 \Rightarrow 6 \Rightarrow 5$ are trivial, and $2 \Rightarrow 4$ as well as $5 \Rightarrow 3$ are clear by the preceding discussion on the reformulation of functions from NPSV as tuples of disjoint NP-sets. It therefore remains to prove the implications $1 \Rightarrow 2$ and $3 \Rightarrow 1$.

For the first of these implications let g be a function many-one complete for NPSV and let $A \subseteq \Sigma^*$ be decidable in polynomial time. We fix some element $a_0 \in A$ and define the function σ as

$$\sigma(y) = \begin{cases} y & y \in A \\ a_0 & \text{otherwise.} \end{cases}$$

As A is decidable in polynomial time, the function σ is in FP. Then $g'(x) = \sigma(g(x))$ is a function in NPSV_A. Observe that for a function $h \in \mathsf{NPSV}_A$ any many-one reduction from h to g also reduces h to g'. Thus g' is many-one complete for NPSV_A.

To prove that item 3 implies item 1, we show that NPSV can be characterized as $\mathsf{FP}^{\mathsf{NPSV}_A}$, where the value $M^f(x)$ computed by the deterministic oracle transducer M on input x is only defined if all oracle queries belong to the domain of the functional oracle f. We first show that $\mathsf{FP}^{\mathsf{NPSV}_A} \subseteq \mathsf{NPSV}$. Clearly, any function in $\mathsf{FP}^{\mathsf{NPSV}_A}$ is single-valued. Also a computation of M^f on input x where $f \in \mathsf{NPSV}_A$ can be simulated by a nondeterministic transducer N that simulates M, and for each query z guesses an accepting path of the nondeterministic transducer that computes f and answers with $f(x)$. This guarantees that $M^f(x) = N(x)$ if all oracle queries of M^f on input x are in the domain of f. If not, then by definition, $M^f(x)$ is undefined, and also set-$N(x) = \emptyset$, i.e., $N(x)$ is undefined. This shows $\mathsf{FP}^{\mathsf{NPSV}_A} \subseteq \mathsf{NPSV}$.

To see that every function $f \in \mathsf{NPSV}$ is in $\mathsf{FP}^{\mathsf{NPSV}_A}$, we fix two distinct elements a_0 and a_1 in A and define the following function $g \in \mathsf{NPSV}_A$:

$$g(z) = \begin{cases} a_j & \text{if } z = 1\langle i, x\rangle \text{ and the } i\text{th bit of } f(x) \text{ is } j, \\ a_0 & \text{if } z = 0\langle l, x\rangle \text{ and } |f(x)| < l, \\ a_1 & \text{if } z = 0\langle l, x\rangle \text{ and } |f(x)| = l. \end{cases}$$

Notice that z is in the domain of g, if $z = 1\langle i, x\rangle$ with some x in the domain of f and $1 \leq i \leq |f(x)|$, or if $z = 0\langle l, x\rangle$ with some x in the domain of f and $l \leq |f(x)|$. Now an oracle transducer M^g computes f as follows. On input x M^g first determines the length l of $f(x)$ by querying $0\langle l, x\rangle$ for $l = 0, 1, \ldots$ until $g(0\langle l, x\rangle) = a_1$ (if x is not in the domain of f, then the first query leads to a reject of M^g, otherwise all the strings queried are in the domain of g). If $l = 0$, then M^g outputs λ, otherwise the output of M^g is the bit string $y_1 \cdots y_l$, where $y_i = 1$ if and only if $g(1\langle i, x\rangle) = a_1$. Therefore M^g computes the function f.

Now the assumption that there is a complete function g for NPSV_A implies $\mathsf{NPSV} = \mathsf{FP}^{\mathsf{NPSV}_A} = \mathsf{FP}^g$, hence also NPSV has a complete function.

The additional claims in the theorem about Turing reductions follow directly from the above proof of $3 \Rightarrow 1$. Namely, assuming the existence of a Turing complete function for NPSV_A, the equality $\mathsf{FP}^{\mathsf{NPSV}_A} = \mathsf{NPSV}$ yields the existence of many-one complete functions for NPSV. □

Additionally, we can get results on tuples obeying less restrictive disjointness conditions. Namely, we call a collection of sets $\{D_i\}_{i \in I}$ *k-disjoint* if $\bigcap_{i \in J} D_i = \emptyset$ for all $J \subseteq I$ such that $\|J\| > k$. For $k = 1$ this is just the usual pairwise

disjointness condition, but for increasing k this leads to successively weaker notions. Reductions are easily generalized to this context, i.e., f strongly reduces $(C_1, \ldots, C_l)$ to $(D_1, \ldots, D_l)$ if f is a many-one polynomial-time reduction from the components C_i to D_i for $i = 1, \ldots, l$.

Similarly as above, there is a direct correspondence between k-disjoint l-tuples of NP-sets and functions from NPkV_A, where NPkV_A denotes all functions from NPMV with $\|\text{set-}f(x)\| \leq k$ and set-$f(x) \subseteq A$ for all $x \in \Sigma^*$. Then we have:

Proposition 6. *For all numbers $l > k \geq 1$, there exist $\leq_s$-complete k-disjoint l-tuples of NP-sets if and only if NPkV_A has many-one complete functions for all subsets $A \subset \Sigma^*$ of size $\|A\| = l$.*

Similarly as in [5] we can show that the question of the existence of complete k-disjoint tuples does not depend on the number of components of the tuple, i.e., for all numbers $l, l' > k \geq 1$, complete k-disjoint l-tuples exist if and only if complete k-disjoint l'-tuples exist.

Instead of considering functions from NPkV_A, it is probably more natural to investigate the function class NPkV, that contains all functions from NPMV such that $\|\text{set-}f(x)\| \leq k$ for all $x \in \Sigma^*$ (cf. [14, 54]). Naik, Rogers, Royer, and Selman [38] showed that with respect to refinements the classes NPkV, $k \geq 1$, form a strict hierarchy (called the output-multiplicity hierarchy), unless the polynomial hierarchy collapses to its second level.

Functions from NPkV correspond to k-disjoint tuples of NP-sets where the number of components is not restricted. Analogously to the implication $1 \Rightarrow 2$ in Theorem 5 we can show the following proposition.

Proposition 7. *If NPkV contains many-one complete functions, then NPkV_A contains many-one complete functions for all polynomial-time decidable sets $A \subseteq \Sigma^*$.*

Fenner, Fortnow, Naik, and Rogers [14] investigated the problem whether total functions in NPkV possess refinements in FP. In particular, they proved that the answer to this question is independent of k, i.e., if $\mathsf{NPkV}_t \subseteq_c \mathsf{FP}$ for some $k \geq 2$, then $\mathsf{NPkV}_t \subseteq_c \mathsf{FP}$ holds for all $k \geq 2$. Here we are interested in the question, whether these function classes contain complete sets. Concerning this problem we can prove:

Theorem 8.

1. *If* TAUT *has an optimal proof system, then for all k,* NPkV *has a many-one complete function.*
2. *Let $g(x)$ be a polynomial-time computable function such that for all $x \in \Sigma^*$ we have $g(x) \leq p(|x|)$ for some polynomial p. Then the existence of optimal proof systems for* TAUT *implies the existence of many-one complete functions for* NPgV.

Proof. The proof follows the general method developed in [28], that amounts to bound the complexity of the promise predicates for NPkV and NPgV. In particular, we have to show that these promise predicates are definable in coNP.

For this let N be an NP transducer. Then the promise that $N(x)$ outputs at most k different values can be defined by the formula

$$\forall y_1 \ldots \forall y_{k+1} \left(\left(\bigwedge_{i=1}^{k+1} y_i \in \text{set-}f(x) \right) \rightarrow \bigvee_{1 \leq i < j \leq k+1} y_i = y_j \right), \tag{1}$$

where f is the NPMV function computed by N. As the premise $\bigwedge_{i=1}^{k+1} y_i \in \text{set-}f(x)$ defines an NP-predicate, the whole formula (1) is a condition in coNP. By choosing suitable polynomial-size nondeterministic circuits for f, we can translate the formula (1) to a sequence of polynomial-size propositional formulas $\theta_n^{k,f}(\bar{p}, \bar{q}, \bar{r})$, which contain propositional variables $\bar{p} = p_1, \ldots, p_n$ for the input x, variables $\bar{q}$ for $y_1, \ldots, y_{k+1}$, as well as auxiliary variables $\bar{r}$ for the gates of the circuits for f.

From the construction of $\theta_n^{k,f}$ it is clear, that f is indeed a function from NPkV if and only if $(\theta_n^{k,f})_{n \geq 0}$ is a sequence of propositional tautologies. As for each NP transducer N the sequence $\theta_n^{k,f}$ can be constructed in polynomial time, we can easily define a proof system h_f which admits polynomial-size proofs of the sequence $\theta_n^{k,f}$. By assumption there exists an optimal proof system h. As h simulates all proof systems h_f, we have polynomial-size h-proofs of $\theta_n^{k,f}$ for all $f \in \mathsf{NPkV}$.

We now claim that the following function f_k is complete for NPkV: f_k takes inputs of the form $\langle x, N, 0^m \rangle$. From this input, f_k first computes the formula $\theta_{|x|}^{k,f}$ where f is the function computed by N. Then f_k guesses an h-proof π of size $\leq m$ and verifies whether $h(\pi) = \theta_{|x|}^{k,f}$. If this is not the case, then f_k stops without producing any output. Otherwise, f_k simulates $N(x)$ for at most m steps and gives the corresponding output. Clearly, f_k belongs to NPkV. To verify its completeness let N be an NP transducer computing a function $f \in \mathsf{NPkV}$ and let p be a polynomial bounding the running time of N as well as the size of h-proofs for $\theta_n^{k,f}$. Then f many-one reduces to f_k via the mapping $x \mapsto \langle x, N, 0^{p(|x| + |\theta_n^{k,f}|)} \rangle$.

For item 2 let $g \in \mathsf{FP}$ such that for all $x \in \Sigma^*$ we have $g(x) \leq p(|x|)$ for some polynomial p. Similarly as above, we define for each function $f \in \mathsf{NPgV}$ the promise of $f(x)$ with respect to NPgV by

$$\forall y_1 \ldots \forall y_{g(x)+1} \left(\left(\bigwedge_{i=1}^{g(x)+1} y_i \in \text{set-}f(x) \right) \rightarrow \bigvee_{1 \leq i < j \leq g(x)+1} y_i = y_j \right). \tag{2}$$

By the conditions on g, the propositional translations of (2) have polynomial size in the length of x. A complete function for NPgV is then obtained analogously as in the proof of item 1. □

Note that if $g(x)$ is a function with super-polynomial increase in $|x|$, then it is not clear whether the formulas (2) can be described by propositional formulas of size polynomial in $|x|$, and therefore the above proof method fails for such functions g. We also leave open, whether the reverse implications of items 1 and

2 are valid. As a more general programme, it seems interesting to determine the relationship between the assumptions of the existence of complete functions in NPkV and NPgV for different numbers k and functions g.

We conclude this section by observing that the class of disjoint coNP-pairs corresponds to the class NPbV_t of all 0,1-valued functions in NPMV_t, studied in [14]. With a disjoint coNP-pair (A_0, A_1) we associate the function $h \in \mathsf{NPbV}_t$ defined by set-$h(x) = \{b \mid x \notin A_b\}$).

Again, it is interesting to see what happens if we extend the range from $\{0,1\}$ in NPbV_t to arbitrary sets $A \subseteq \Sigma^*$. If $A = \{a_1, \ldots, a_k\}$ contains exactly k elements, then a function g from $\mathsf{NPMV}_{t,A}$ corresponds to a tuple $(A_1, \ldots, A_k)$ of coNP-sets with $A_i = \{x \mid a_i \notin \text{set-}g(x)\}$. As every x can be contained in at most $k-1$ sets from $A_1, \ldots, A_k$, the tuple $(A_1, \ldots, A_k)$ is $(k-1)$-disjoint (but not necessarily pairwise disjoint). Given this correspondence between coNP-tuples and functions from NPMV_t, we obtain the following result.

Theorem 9.

1. *If* NPMV_t *has many-one complete functions, then there exist strongly many-one complete disjoint* coNP*-pairs.*
2. *More generally, if* NPMV_t *has many-one complete functions, then there exist* $\leq_s$*-complete* $(k-1)$*-disjoint* k*-tuples of* coNP*-sets for all* $k \geq 2$*.*

Proof. It suffices to prove the second item. Using a similar argument as for the implication $1 \Rightarrow 2$ in Theorem 5, we can show that the existence of complete functions for NPMV_t implies that for every $k \geq 2$ there exist complete functions in $\mathsf{NPMV}_{t,A}$ for each $A \subset \Sigma^*$ containing exactly k elements. By the above correspondence between functions from $\mathsf{NPMV}_{t,A}$ and $(k-1)$-disjoint k-tuples of coNP-sets, we obtain the asserted complete k-tuple. □

We leave open whether the reverse implications also hold.

4 Collapse of NPSV and Effective Interpolation

In this section we investigate the question whether functions from NPSV admit total extensions in FP or FP/poly. We show that this question can be characterized by interpolation properties, which in turn are intimately connected with disjoint NP-pairs associated with propositional proof systems. We will start by reviewing different notions of interpolation along with their connections to disjoint NP-pairs.

Due to Craig's interpolation theorem for propositional logic, for any tautology $\varphi \to \psi$ there is a formula θ that uses only common variables of φ and ψ such that $\varphi \to \theta$ and $\theta \to \psi$ are tautologies [12]. A circuit C that computes the same function as θ is called an *interpolant* of $\varphi \to \psi$.

Lower bounds for the size of interpolants were first considered by Mundici [37], who proved that the existence of polynomial-size interpolants for all tautologies $\varphi \to \psi$ implies $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$. As the existence of polynomial-size interpolants for all tautological implications seems to be a rather strong assumption, Krajíček [31] suggested to measure the size of an interpolant not merely

in terms of the size of the implication $\varphi \to \psi$, but in terms of the size of the shortest proof of this implication in some fixed proof system. This leads to the following definition:

Definition 10 (Krajíček, Pudlák [33]). *A proof system h for* TAUT *admits* effective interpolation *if there is a polynomial p such that for any h-proof w of a formula $h(w) = \varphi \to \psi$, the formula $h(w)$ has an interpolant of size at most $p(|w|)$.*

Effective interpolation is sometimes considered in an efficient version such that it is possible to generate an interpolating circuit from an h-proof of a formula $\varphi \to \psi$ in polynomial time. In [42] this property is called *feasible interpolation*.

Feasible interpolation has been shown for resolution [31], the cutting planes system [7, 31, 40], and some algebraic proof systems [43]. Combined with lower bounds for the separation of the clique colouring pair by monotone Boolean circuits [44, 1], these results yield lower bounds for the proof lengths in the above proof systems. We refer to the survey [41] for a detailed presentation of this approach.

The notion of effective interpolation for a propositional proof system h can be characterized by a disjoint NP-pair associated with the proof system h. For this we define the following *interpolation pair* $Int(h)$ with the components

$$\begin{aligned} Int_1(h) = \{\langle \varphi(\bar{x},\bar{y}), \psi(\bar{x},\bar{z}), \bar{a}, 0^m\rangle \mid\ & \bar{x} \text{ are the common variables of } \varphi \text{ and } \psi, \\ & \varphi(\bar{a},\bar{y}) \text{ is satisfiable, and} \\ & h \vdash_{\le m} \varphi(\bar{x},\bar{y}) \to \psi(\bar{x},\bar{z})\} \\ Int_2(h) = \{\langle \varphi(\bar{x},\bar{y}), \psi(\bar{x},\bar{z}), \bar{a}, 0^m\rangle \mid\ & \bar{x} \text{ are the common variables of } \varphi \text{ and } \psi, \\ & \neg\psi(\bar{a},\bar{z}) \text{ is satisfiable, and} \\ & h \vdash_{\le m} \varphi(\bar{x},\bar{y}) \to \psi(\bar{x},\bar{z})\}. \end{aligned}$$

Let us first argue that $Int(h)$ is indeed a disjoint NP-pair. Clearly, both components are in NP. To verify the disjointness, assume that $\langle \varphi(\bar{x},\bar{y}), \psi(\bar{x},\bar{z}), \bar{a}, 0^m\rangle$ is contained in $Int_1(h)$. Since we have an h-proof, the formula $\varphi(\bar{x},\bar{y}) \to \psi(\bar{x},\bar{z})$ is a tautology. By assumption, $\varphi(\bar{a},\bar{y})$ is satisfiable and hence $\psi(\bar{a},\bar{z})$ must be a tautology. Therefore, $\neg\psi(\bar{a},\bar{z})$ is unsatisfiable which implies $\langle \varphi(\bar{x},\bar{y}), \psi(\bar{x},\bar{z}), a, 0^m\rangle \notin Int_2(h)$.

Before we start to explain the link of the interpolation pair to different notions of interpolation, let us mention a general connection between propositional proof systems and disjoint NP-pairs. For this connection, disjoint NP-pairs are represented by sequences of propositional formulas (cf. [3]). The formal definition is as follows: A *propositional representation* for an NP-set A is a sequence of propositional formulas $\varphi_n(\bar{x},\bar{y})$ with the following properties:

1. $\varphi_n(\bar{x},\bar{y})$ has propositional variables $\bar{x}$ and $\bar{y}$ such that $\bar{x}$ is a vector of n propositional variables.
2. There exists a polynomial-time algorithm that on input 1^n outputs $\varphi_n(\bar{x},\bar{y})$.
3. Let $\bar{a} \in \{0,1\}^n$. Then $\bar{a} \in A$ if and only if $\varphi_n(\bar{a},\bar{y})$ is satisfiable (where we have substituted the propositional variables $\bar{x}$ by the bits $\bar{a}$).

With these propositional descriptions of NP-sets we can represent disjoint NP-pairs in propositional proof systems. We say that a disjoint NP-pair (A, B) is *representable in a propositional proof system* h if there are propositional representations $\varphi_n(\bar{x}, \bar{y})$ of A and $\psi_n(\bar{x}, \bar{z})$ of B such that $\bar{x}$ are the common variables of $\varphi_n(\bar{x}, \bar{y})$ and $\psi_n(\bar{x}, \bar{z})$ and $h \vdash_{\leq p(n)} \varphi_n(\bar{x}, \bar{y}) \to \neg\psi_n(\bar{x}, \bar{z})$ for some polynomial p.

Let us remark at this point that every disjoint NP-pair (A, B) is representable in some propositional proof system by simply coding a representation of (A, B) into a given base system. As a concrete example, let us explain how this works for the *extended Frege proof system EF* (cf. [11]). If $\varphi_n(\bar{x}, \bar{y})$ and $\psi_n(\bar{x}, \bar{z})$ are propositional representations for the NP-sets A and B, respectively, then the pair (A, B) is representable in the system $EF + \{\varphi_n(\bar{x}, \bar{y}) \to \neg\psi_n(\bar{x}, \bar{z}) \mid n \geq 0\}$ which augments EF by additional axioms for the disjointness of the pair (A, B). Such extensions $EF + \Phi$ by polynomial-time decidable sets Φ of tautologies are of particular interest, as every propositional proof system is simulated by such a system $EF + \Phi$ for suitable axioms Φ (cf. [30]).

Before we explain how the interpolation pair $Int(h)$ captures effective interpolation for h, we will show that $Int(h)$ serves as a hard pair for the class of all pairs representable in the system h (this class was investigated in detail in [3] under the name DNPP(h)). We formulate this observation in the next proposition.

Proposition 11. *For every proof system h the interpolation pair $Int(h)$ is $\leq_s$-hard for the class of all disjoint* NP*-pairs that are representable in h.*

Proof. Let h be a proof system and let (A, B) be a disjoint NP-pair such that $\varphi_n(\bar{x}, \bar{y})$ and $\psi_n(\bar{x}, \bar{z})$ represent A and B, respectively, and $h \vdash_{\leq p(n)} \varphi_n(\bar{x}, \bar{y}) \to \neg\psi_n(\bar{x}, \bar{z})$ for some polynomial p. It is then straightforward to verify that

$$a \mapsto \langle \varphi_{|a|}(\bar{x}, \bar{y}), \neg\psi_{|a|}(\bar{x}, \bar{z}), a, 0^{p(|a|)} \rangle$$

realizes the reduction $(A, B) \leq_s Int(h)$. □

Now we want to argue that $Int(h)$ indeed justifies its qualification as a pair that describes the effective interpolation property. To this end we consider for a given proof system h the following three assertions:

$A_1(h)$: The interpolation pair $Int(h)$ is P/poly-separable.
$A_2(h)$: h has effective interpolation.
$A_3(h)$: All NP-pairs that are representable in h are P/poly-separable.

Then the following implications between these assertions hold.

Proposition 12.

1. *For all propositional proof systems h the implications $A_1(h) \Rightarrow A_2(h) \Rightarrow A_3(h)$ hold.*
2. *Let h be a proof system of the form $EF + \Phi$ with a polynomial-time decidable set of tautologies Φ. Then the equivalences $A_1(h) \Leftrightarrow A_2(h) \Leftrightarrow A_3(h)$ hold.*

Proof. To prove the implication $A_1(h) \Rightarrow A_2(h)$ for arbitrary proof systems h, assume that $Int(h)$ is separated by the polynomial-size circuit family C_n, i.e., for inputs $\langle \varphi(\bar{x},\bar{y}), \psi(\bar{x},\bar{z}), a, 0^m\rangle$ of length n from $Int_1(h)$ the circuit C_n outputs 1, and C_n outputs 0 for inputs from $Int_2(h)$.

Let $\varphi(\bar{x},\bar{y}) \to \psi(\bar{x},\bar{z})$ be an implication that has an h-proof of length m. By substituting φ, ψ, and 0^m for the respective input gates of the appropriate circuit C_n, we obtain a circuit with inputs $\bar{x}$ that interpolates φ and ψ.

For the implication $A_2(h) \Rightarrow A_3(h)$ let (A, B) be a disjoint NP-pair that is representable in h with respect to the representations $\varphi_n(\bar{x},\bar{y})$ and $\psi_n(\bar{x},\bar{z})$, i.e., we have h-proofs of length $\leq p(n)$ for the sequence of formulas $\varphi_n(\bar{x},\bar{y}) \to \neg\psi_n(\bar{x},\bar{z})$ with some polynomial p. As h has effective interpolation by $A_2(h)$, there exist interpolating circuits $C_n(\bar{x})$ for $\varphi_n(\bar{x},\bar{y}) \to \neg\psi_n(\bar{x},\bar{z})$. Hence the circuit family C_n provides a separator for (A, B).

For item 2 it remains to show the implication $A_3(h) \Rightarrow A_1(h)$ for proof systems h of the form $EF + \Phi$ with polynomial-time decidable set $\Phi \subseteq \mathrm{TAUT}$. For this it suffices to prove the representability of $Int(h)$ in the system h, i.e., we have to construct representations of $Int(h)$ such that h admits short proofs for the disjointness of $Int_1(h)$ and $Int_2(h)$ with respect to these representations. Such representations arise from translations of natural arithmetic formulas defining the components of $Int(h)$. Using reflection for extensions of EF, it is then straightforward to verify the disjointness of $Int(h)$ in the system h (cf. [3, 5] for a detailed description of this procedure). □

To capture the feasible interpolation property, Pudlák [42] defines an *interpolation pair* (I_h^0, I_h^1) for a proof system h with the components

$$I_h^i = \{\langle \varphi_0, \varphi_1, \pi\rangle \mid \varphi_0 \text{ and } \varphi_1 \text{ do not share variables, } \neg\varphi_i \text{ is satisfiable, and } h(\pi) = \varphi_0 \vee \varphi_1\}$$

for $i = 0, 1$. Let us briefly argue for the disjointness of the pair: The proof π ensures that $\varphi_0 \vee \varphi_1$ is tautological and since φ_0 and φ_1 do not share variables, one of the formulas φ_0 or φ_1 must itself be a tautology. Therefore, $\neg\varphi_0$ and $\neg\varphi_1$ cannot be both satisfiable and hence $I_h^0 \cap I_h^1 = \emptyset$. Under reasonable assumptions on the proof system h, we can then show a similar result as in Proposition 12 for (I_h^0, I_h^1) and the efficient analogues of $A_1(h)$ to $A_3(h)$. In particular, h has feasible interpolation if and only if (I_h^0, I_h^1) is P-separable (assuming some simple closure properties of h such as closure under substitution by constants, cf. [42]). The pair (I_h^0, I_h^1) is, however, not suitable for the notion of effective interpolation, for which reason we have defined its nonuniform version $Int(h)$.

As mentioned above, weak systems like resolution or cutting planes are known to possess effective interpolation [7, 31, 40]. In contrast, there is evidence that strong propositional proof systems like Frege systems and their extensions do not admit effective interpolation [33, 8, 6]. In particular, it is observed in [33] that extended Frege proof systems do not admit effective interpolation if the RSA cryptosystem is secure.

Partly generalizing this observation, one can state that the existence of an honest injective function in FP that is not $\mathsf{FP/poly}$-invertible (i.e., a one-way

function that is secure against FP/poly) implies the existence of a proof system for TAUT that does not admit effective interpolation. Notice that each injective function in FP is invertible by an NPSV-function. Thus the assumption that each NPSV-function has a total extension in FP/poly implies that every injective function is FP/poly-invertible. As the former assumption implies $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$ and the latter is equivalent to $\mathsf{UP} \subseteq \mathsf{P/poly}$ (cf. [26, 22]), the former assumption is presumably stronger. We now show that every function in NPSV has a total extension in FP/poly if and only if every proof system for TAUT admits effective interpolation.

Theorem 13. *The following statements are equivalent.*

1. *Every propositional proof system admits effective interpolation.*
2. *Every disjoint* NP*-pair is* P/poly*-separable.*
3. *Every function in* NPSV *has a total extension in* FP/poly*.*
4. *For every set* $S \subseteq$ TAUT*,* $S \in \mathsf{NP}$*, there is a polynomial* p*, such that any formula* $\varphi \to \psi \in S$ *has an interpolant of size at most* $p(|\varphi \to \psi|)$*.*
5. *For every printable set* $S \subseteq$ TAUT*, there is a polynomial* p*, such that any formula* $\varphi \to \psi \in S$ *has an interpolant of size at most* $p(|\varphi \to \psi|)$*.*

Proof. For the proof we will show the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5$ as well as $4 \Rightarrow 1$ and $5 \Rightarrow 2$. The implication $4 \Rightarrow 5$ is immediate, as item 5 is a weakening of item 4.

Items 1 and 2 are the universally quantified versions of the assertions $A_2(h)$ and $A_3(h)$, respectively, i.e., item 1 expresses that $A_2(h)$ holds for all propositional proof systems h. Similarly, this holds for item 2 and assertion $A_3(h)$, as every disjoint NP-pair (A, B) is representable in a proof system $EF + \{\varphi_n \to \neg\psi_n\}$ with arbitrary representations φ_n and ψ_n for A and B, respectively. Therefore the equivalence of items 1 and 2 is a direct consequence of Proposition 12.

The implication $2 \Rightarrow 3$ was shown in [52], but for the sake of completeness we include a proof. Assume that all disjoint NP-pairs are P/poly-separable, and let f be a function in NPSV. With f we associate a pair (A_0^f, A_1^f) with the components

$$A_i^f = \{\langle x, j\rangle \mid (\exists y) y \in \text{set-}f(x),\ 1 \le j \le |y|,\ \text{and the } j\text{-th bit of } y \text{ is } i\}.$$

This disjoint NP-pair describes all bits of the values of f. To determine the length of f-values we define a second NP-pair (B_0^f, B_1^f) with the components

$$B_0^f = \{\langle x, j\rangle \mid (\exists y) y \in \text{set-}f(x) \text{ and } j \le |y|\}$$
$$B_1^f = \{\langle x, j\rangle \mid (\exists y) y \in \text{set-}f(x) \text{ and } j > |y|\}.$$

By assumption the pairs (A_0^f, A_1^f) and (B_0^f, B_1^f) can be separated by polynomial-size circuit families C_n and D_n, respectively. Using these circuits we devise a function $g \in \mathsf{FP/poly}$ that refines f as follows. Let p be a polynomial bounding the running time of f. At input x, the function g evaluates all respective circuits from D_n on inputs $\langle x, 1\rangle, \ldots, \langle x, p(|x|) + 1\rangle$ to determine the length l of the possible output value of $f(x)$. After l is computed, g evaluates the circuits C_n

on inputs $\langle x, 1\rangle, \ldots, \langle x, l\rangle$. The output of g is then just the bitwise concatenation of these values. From the construction it is clear that $g \in \mathsf{FP/poly}$ refines the function f.

The proof of the implication $3 \Rightarrow 4$ is obtained by extending an idea from [52]. Let $S \subseteq \mathrm{TAUT}$, $S \in \mathsf{NP}$. Let f be a function such that for any formula $\varphi \in S$, $\varphi = \varphi_0(\bar{x}, \bar{y}) \to \varphi_1(\bar{x}, \bar{z})$, it holds

$$f(\langle \alpha, \varphi \rangle) = \begin{cases} 1 & \text{if for some } \beta, \quad \varphi_0(\alpha, \beta) \text{ holds} \\ 0 & \text{if for some } \gamma, \ \neg\varphi_1(\alpha, \gamma) \text{ holds.} \end{cases}$$

Otherwise, and for any other input let f be undefined. First observe that f is well defined, i.e., that f is single valued. This is due to the fact that $\varphi = \varphi_0(\bar{x}, \bar{y}) \to \varphi_1(\bar{x}, \bar{z}) \in \mathrm{TAUT}$. Further, f can be computed by a nondeterministic machine N that first (in deterministic polynomial time) validates that the input is of the appropriate form $\langle \alpha, \varphi \rangle$, $\varphi = \varphi_0(\bar{x}, \bar{y}) \to \varphi_1(\bar{x}, \bar{z})$. Then N guesses a certificate for $\varphi \in S$ and, if successful, guesses some string w. Now if w is of an appropriate length and if $\varphi_0(\alpha, w)$ holds, then N outputs 1, if $\varphi_1(\alpha, w)$ holds, N outputs 0. Hence $f \in \mathsf{NPSV}$.

Assuming 3, f has a total extension in $\mathsf{FP/poly}$. Thus there is a polynomial p and for any $n \geq 0$ a circuit C_n of size at most $p(n)$ such that for any tuple $v = \langle \alpha, \varphi \rangle$ of length n in the domain of f, $C_n(v) = f(v)$. Fixing the input bits of C_n that belong to the formula φ, we obtain a circuit C_φ with $C_\varphi(\alpha) = C_n(\langle \alpha, \varphi \rangle) = f(\langle \alpha, \varphi \rangle)$, and thus C_φ is of size polynomial in $|\varphi|$. Now observe that C_φ is an interpolant for the formulas $\varphi_0(\bar{x}, \bar{y})$ and $\varphi_1(\bar{x}, \bar{z})$. If $\varphi_0(\alpha, \bar{y}) \in \mathrm{SAT}$, then $C_\varphi(\alpha) = 1$, and if $C_\varphi(\alpha) = 1$, then for no γ it holds $\neg\varphi_1(\alpha, \gamma)$ and therefore $\varphi_1(\alpha, \bar{z}) \in \mathrm{TAUT}$.

To prove the implication $4 \Rightarrow 1$, let pad: $\Sigma^* \times \{0\}^* \to \Sigma^*$ be a function in FP with the following properties:

1. $\mathrm{pad}(\chi, 0^n) \in \mathrm{TAUT}$ if and only if $\chi \in \mathrm{TAUT}$.
2. given an implication $\varphi \to \psi \in \mathrm{TAUT}$ as an input, the output $\mathrm{pad}(\varphi \to \psi, 0^n)$ is also an implication $\varphi' \to \psi'$ that has the same interpolants as $\varphi \to \psi$.
3. $|\mathrm{pad}(\chi, 0^n)| \geq |\chi| + n$.

Notice that there is such a padding function. Now let h be a proof system for TAUT, and let

$$S = \{\chi \mid \exists n \leq |\chi| \, \exists w, |w| \leq n, \ \mathrm{pad}(h(w), 0^n) = \chi\}.$$

Clearly $S \in \mathsf{NP}$, as pad and h are functions in FP. Because h is a proof system for TAUT and due to property 1 of pad, $S \subseteq \mathrm{TAUT}$. Thus by assumption 4 there is a monotone polynomial p, such that any formula $\varphi' \to \psi' \in S$ has an interpolant of size at most $p(|\varphi' \to \psi'|)$. As pad, $h \in \mathsf{FP}$ there are monotone polynomials q, r such that $|h(w)| \leq q(|w|)$ and $|\mathrm{pad}(\chi, 0^n)| \leq r(|\chi| + n)$ for any $w, \chi \in \Sigma^*$, $n \geq 0$. Let $\varphi \to \psi \in \mathrm{TAUT}$ and let w be an h-proof for $\varphi \to \psi$. Now by property 3 of pad $\varphi' \to \psi' = \mathrm{pad}(\varphi \to \psi, 0^{|w|}) \in S$, therefore by the property 2 of pad, $\varphi \to \psi$ has an interpolant of size at most $p(|\varphi' \to \psi'|) \leq p(r(q(|w|) + |w|))$.

To finish the proof let us show that the implication $5 \Rightarrow 2$ holds. Let (A, B) be a disjoint NP-pair. We choose arbitrary representations $\varphi_n(\bar{x}, \bar{y})$ for A and $\psi_n(\bar{x}, \bar{z})$ for B. By the disjointness of A and B, $\varphi_n(\bar{x}, \bar{y}) \to \neg\psi_n(\bar{x}, \bar{z})$ is a printable sequence of tautologies. Assuming 5 we get polynomial-size interpolating circuits for these formulas. These circuits separate the pair (A, B). □

Part of the equivalences of the last theorem were already shown by Schöning and Torán [52]. There they proved that items 2, 3, and 5 are equivalent, and that these hypotheses imply $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{P/poly}$ and $\mathsf{UP} \subseteq \mathsf{P/poly}$.

Let us note that Theorem 13 also holds in an efficient version, where $\mathsf{FP/poly}$ is replaced by FP, and effective interpolation is strengthened to feasible interpolation. It is readily checked that the proof of Theorem 13 is easily modified to this efficient context. Hence Theorem 13 along with its proof yield the following corollary:

Corollary 14. *The following statements are equivalent.*

1. *Every propositional proof system admits feasible interpolation.*
2. *Every disjoint* NP*-pair is* P*-separable.*
3. *Every function in* NPSV *has a total extension in* FP*.*

Let us mention that the above list of equivalences also relates to the important concept of automatizability, as recently noted by Sadowski [49]. In [8] a proof system h is called *automatizable* if there exists a deterministic procedure that takes as input a formula φ and outputs an h-proof of φ in time polynomial in the length of the shortest h-proof of φ. A proof system g is called *weakly automatizable* if there exists an automatizable system h that simulates g (cf. [42]). In [49] Sadowski proves that items 2 and 3 from Corollary 14 are equivalent to the statement that every propositional proof system is weakly automatizable.

It is easy to see that a proof system g admits effective interpolation if g is simulated by a proof system h that admits effective interpolation. As a corollary from Theorem 13 we obtain:

Corollary 15. *If there is an optimal proof system for* TAUT *that admits effective interpolation, then items 1 to 5 from Theorem 13 hold.*

5 Is the Standard Proof System for SAT P-optimal?

In this section we will consider the question whether the standard proof system for SAT is p-optimal, where by the standard proof system *sat* for SAT we mean the following procedure of checking the truth value of a given assignment:

$$sat(x) = \begin{cases} \varphi & \text{if } x = \langle \alpha, \varphi \rangle \text{ and } \alpha \text{ is a satisfying assignment for } \varphi \\ \text{undef.} & \text{otherwise.} \end{cases}$$

As *sat* is polynomially bounded (i.e. every satisfiable formula has a polynomial-size proof in *sat*), the system *sat* is an optimal proof system for SAT. It will turn out that the question whether *sat* is even p-optimal is (in some disguise) actually well studied in the literature. The assumption that *sat* is p-optimal is

equivalent to a variety of complexity-theoretic assumptions (which have unlikely collapse consequences such as $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$).

In [14] the following statements were all shown to be equivalent. There, Q is defined to be the proposition that one (and consequently each) of these statements is true. In this section we show that Q is also equivalent to the p-optimality of *sat*.

Theorem 16 (Fenner, Fortnow, Naik, Rogers [14]). *The following statements are equivalent.*

1. *For each NPTM N that accepts* SAT *there is a function $f \in \mathsf{FP}$ such that for each accepting path α of N on input φ, $f(\langle \varphi, \alpha \rangle)$ is a satisfying assignment of φ.*
2. *Each honest function $f \in \mathsf{FP}$ with range Σ^* is FP-invertible.*
3. $\mathsf{NPMV}_t \subseteq_c \mathsf{FP}$.
4. *For each P-subset S of* SAT *there exists a function $g \in \mathsf{FP}$ such that for all $\varphi \in S$, $g(\varphi)$ is a satisfying assignment of φ.*

Clearly, each nondeterministic Turing machine N corresponds to a proof system h for SAT with $h(w) = \varphi$ if w encodes an accepting path of N on input φ. Now h is honest if and only if N is a NPTM. This leads to the observation that Statement 1 in Theorem 16 is equivalent to the condition that *sat* p-simulates every proof system h for SAT where h happens to be an honest function. Hence, we just need to delete the term 'polynomial-time' in the Statement 1 of Theorem 16 to obtain the desired result that Q is equivalent to the p-optimality of *sat*. That this is possible without changing the truth of the theorem is shown by a padding argument.

Theorem 17. *The following statements are equivalent.*

1. *For each nondeterministic Turing machine N that accepts* SAT *there is a function $f \in \mathsf{FP}$ such that for each accepting path α of N on input φ, $f(\langle \varphi, \alpha \rangle)$ is a satisfying assignment of φ.*
2. *For each NPTM N that accepts* SAT *there is a function $f \in \mathsf{FP}$ such that for each accepting path α of N on input φ, $f(\langle \varphi, \alpha \rangle)$ is a satisfying assignment of φ.*
3. *sat is a p-optimal proof system for* SAT.

Proof. By the preceding discussion, it is clear that items 1 and 3 are equivalent. Also, item 1 trivially implies item 2. Hence it remains to prove that 2 implies 3.

For this assume that item 2 holds, and let h be a proof system for SAT. We will show that *sat* p-simulates h. Let $\top$ be some tautology, and let $\top^1 = \top$, $\top^n = \top \wedge \top^{n-1}$ for $n \geq 2$ (i.e., $\top^n$ is a tautology of length $\geq n$ that is easy to compute from 0^n). Let $h' \in \mathsf{FP}$ be a proof system defined by

$$h'(x) = \begin{cases} \varphi \wedge \top^{|x|-1} & \text{if } x = 1w \text{ and } h(w) = \varphi, \\ \varphi & \text{if } x = 0w \text{ and } sat(w) = \varphi, \\ \text{undef.} & \text{otherwise.} \end{cases}$$

Notice that h' is honest. Hence, *sat* p-simulates h' by the following argument. Let $N_{h'}$ be a nondeterministic polynomial-time Turing machine that on input y guesses some w and accepts if $h'(w) = y$. Since $N_{h'}$ accepts SAT, by item 2 there is a function $f \in \mathsf{FP}$ such $f(\langle y, \alpha \rangle)$ is a satisfying assignment of y for any accepting path α of $N_{h'}$ on input y. Thus there is a function $f' \in \mathsf{FP}$ with $sat(f'(w)) = \varphi \wedge \top^{|w|}$ for any w with $h(w) = \varphi$. But it is clear that from a satisfying assignment of $\varphi \wedge \top^{|w|}$ we can easily compute a satisfying assignment of φ, i.e., there is a function g with $sat(g(f'(w), h(w))) = h(w)$ for any w in the domain of h. □

It is known that the assumption $\mathsf{NP} = \mathsf{P}$ implies $\mathsf{NPMV}_t \subseteq_c \mathsf{FP}$ which in turn implies $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}$ (cf. [55]). Also, in [25] it has been shown that the converse of these implications is not true in suitable relativized worlds. The consequence $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}$ also shows that the assumption that *sat* is p-optimal is presumably stronger than the assumption that SAT has a p-optimal proof system. Namely the p-optimality of *sat* implies that $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}$, whereas the existence of a p-optimal proof system follows already if any super-tally set in Σ_2^P belongs to P, where any set $L \subseteq \{0^{2^{2^n}} \mid n \geq 0\}$ is called super-tally [28].

The assumption that *sat* is a p-optimal proof system also has an effect on various reducibility degrees, as has been mentioned in [14] for Karp and Levin reducibility. Also in [35] it is shown that $\mathsf{NPMV}_t \subseteq_c \mathsf{FP}$ if and only if γ-reducibility equals polynomial-time many-one reducibility. Furthermore it is shown in [13] that Statement 4 of Theorem 16 is equivalent to the assumption that the approximation class APX is closed under L-reducibility (see [13] for definitions).

The equivalence between the p-optimality of *sat* and $\mathsf{NPMV}_t \subseteq_c \mathsf{FP}$ directly leads to the following theorem.

Theorem 18. *The following statements are equivalent.*

1. *sat is p-optimal.*
2. *For all languages the notions of simulation and p-simulation coincide, i.e., for every language L and all proof systems h and g for L we have $g \leq h$ if and only if $g \leq_p h$.*
3. *The notions of simulation and p-simulation coincide for propositional proof systems, i.e., for all propositional proof systems h and g we have $g \leq h$ if and only if $g \leq_p h$.*
4. *Every optimal proof system is p-optimal.*

Proof. We will show the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$ and $2 \Rightarrow 4 \Rightarrow 1$, of which $2 \Rightarrow 3$ and $2 \Rightarrow 4$ are obvious, and $4 \Rightarrow 1$ follows, because *sat* is optimal.

To show the implication $1 \Rightarrow 2$, assume that *sat* is a p-optimal proof system. Clearly, if h p-simulates g, then h also simulates g.

Therefore, let us assume that h simulates g. Then there is a polynomial p such that for every x in the domain of g there is some w of length at most $p(|x|)$ with $g(x) = h(w)$. Let N be a nondeterministic polynomial-time Turing machine that on input of any x in the domain of g guesses some w of length $\leq p(|x|)$ and outputs w if $h(w) = g(x)$ (if x is not in the domain of g, which can be decided in

polynomial time, N outputs λ). Clearly, N computes a function in NPMV_t. As by assumption *sat* is p-optimal, we get $\mathsf{NPMV}_t \subseteq_c \mathsf{FP}$ by Theorems 16 and 17. Hence there exists a function $f \in \mathsf{FP}$ with $h(f(x)) = g(x)$ for every x in the domain of g. Therefore h p-simulates g.

Now we prove the remaining implication $3 \Rightarrow 1$. For this let g be an arbitrary propositional proof system, and let N be an NPTM for SAT. From g and N we define a propositional proof system h_1 by

$$h_1(\pi) = \begin{cases} g(\pi') & \text{if } \pi = 0\pi', \text{ and } g(\pi') \text{ is not of the form } \top \vee \varphi \\ & \quad \text{with some propositional formula } \varphi \\ \top \vee \varphi & \text{if } \pi = 1\langle \varphi, 1^m \rangle \text{ and } m \geq 2^{|\varphi|} \\ \top \vee \varphi & \text{if } \pi = 1\langle \varphi, w \rangle \text{ and } w \text{ is an accepting path of } N(\varphi) \\ \top & \text{otherwise,} \end{cases}$$

where $\top$ stands for a fixed tautology. Clearly, h_1 is computable in polynomial time and outputs only tautologies. Moreover, all tautologies appear in the range of h_1, according to the first two lines of its definition. Hence h_1 is a propositional proof system.

Similarly, we construct a propositional proof system h_2 by replacing N by *sat* in the third line of the definition of h_1.

$$h_2(\pi) = \begin{cases} g(\pi') & \text{if } \pi = 0\pi', \text{ and } g(\pi') \text{ is not of the form } \top \vee \varphi \\ & \quad \text{with some propositional formula } \varphi \\ \top \vee \varphi & \text{if } \pi = 1\langle \varphi, 1^m \rangle \text{ and } m \geq 2^{|\varphi|} \\ \top \vee \varphi & \text{if } \pi = 1\langle \varphi, \alpha \rangle \text{ and } \alpha \text{ is a satisfying assignment for } \varphi \\ \top & \text{otherwise.} \end{cases}$$

Obviously, h_1 and h_2 are equivalent, as they differ only in proofs for formulas $\top \vee \varphi$ with $\varphi \in \mathrm{SAT}$, and these tautologies have polynomial-size proofs in both h_1 and h_2 (these formulas also have exponential-size proofs in both systems). Thus, assuming 3, h_1 and h_2 are p-equivalent. Let $f \in \mathsf{FP}$ compute a p-simulation of h_1 by h_2, and let w be an accepting path of N on input φ. Then f computes on input $1\langle \varphi, w \rangle$ a satisfying assignment α for φ (the complete output of f is $1\langle \varphi, \alpha \rangle$). Thus assertion 2 of Theorem 17 holds, which we have already shown to be equivalent to the p-optimality of *sat* in Theorem 17. □

The equivalence $2 \Leftrightarrow 3$ of the preceding theorem states that any simulation of a propositional proof system can be turned into a p-simulation if and only if any simulation of an arbitrary proof system can be turned into a p-simulation. In contrast, we cannot expect a similar equivalence with respect to the existence of optimal and p-optimal proof systems since item 4 of the previous theorem is probably stronger than the statement that every optimal propositional proof system is p-optimal. The reason for this is that optimal propositional proof systems are unlikely to exist (cf. [28]). Therefore, item 4 restricted to propositional proof systems would be trivially true, whereas item 4 is probably false, as it is equivalent to Q and hence leads to unlikely collapse consequences.

In [36] it has been observed that given a p-optimal proof system h for a language L, the problem to find an h-proof for $y \in L$ is not much harder than deciding L. More precisely, we can transform each deterministic Turing machine M with $L(M) = L$ to a deterministic Turing machine M' that on input $y \in L$ yields an h-proof of y in $\text{time}_{M'}(y) \leq p(|y| + \text{time}_M(y))$, for some polynomial p determined by M. Using this observation and the equivalences in Theorems 16 and 17 we obtain the following result: *sat* is p-optimal if and only if any deterministic Turing machine M that accepts SAT can be converted to a deterministic Turing machine that computes a satisfying assignment for any formula $\varphi \in \text{SAT}$ and runs not much longer than M on input φ.

Theorem 19. *The following statements are equivalent.*

1. *sat is p-optimal.*
2. *For any deterministic Turing machine M that accepts* SAT *in $\text{time}_M(\varphi)$ steps for any $\varphi \in \text{SAT}$, there is a deterministic Turing machine M' and a polynomial p such that for every $\varphi \in \text{SAT}$, M' produces a satisfying assignment of φ in $\text{time}_{M'}(\varphi) \leq p(|\varphi| + \text{time}_M(\varphi))$ steps.*

Proof. It is easy to see that *2* implies Statement 4 of Theorem 16: Combine a polynomial-time machine that decides a P-subset S of SAT with a standard machine that decides SAT to obtain a machine M for SAT whose running time $\text{time}_M(\varphi)$ is polynomial in $|\varphi|$ for $\varphi \in S$. Assuming *2* there is a machine M' that on input $\varphi \in S$ produces a satisfying assignment of φ in time polynomial in $|\varphi|$.

To show the implication *1* $\Rightarrow$ *2* assume that *sat* is p-optimal. Let M be an arbitrary deterministic Turing machine that decides SAT. We will construct a suitable machine M' to show that *2* holds. Define a proof system h_M for SAT with $h_M(\langle \varphi, 0^s \rangle) = \varphi$ if M accepts φ in at most s steps. As *sat* is p-optimal there is a function $g \in \mathsf{FP}$ with $sat(g(x)) = h_M(x)$ for each x in the domain of h_M. Now on input φ, M' simulates M. If M accepts after s steps, M' computes the *sat*-proof $g(\langle \varphi, 0^s \rangle)$ of φ and extracts the satisfying assignment. Otherwise M' rejects. Clearly, on input $\varphi \in \text{SAT}$, M' needs time at most polynomial in $|\varphi| + \text{time}_M(\varphi)$. □

Under the assumption that *sat* is not p-optimal it follows from Theorem 19 that there is a Turing machine M that decides SAT such that any machine M' that on input $\varphi \in \text{SAT}$ has to produce a satisfying assignment for φ is much slower on some SAT instances. In some sense this appears counter-intuitive as probably all SAT algorithms used in practice produce a satisfying assignment in case the input belongs to SAT. Of course it follows from Theorem 19 that M is superior to any such M' on an infinite set of instances. As shown in the following theorem M is even superior to any M' on a fixed non-sparse set of SAT instances. The result is due to the paddability of SAT, and uses ideas from the theory of complexity cores (cf. [51]).

Theorem 20. *The following statements are equivalent.*

1. *sat is not p-optimal.*

2. *There is a* P*-subset* S *of* SAT *such that for any deterministic Turing machine* M' *that on input* $\varphi \in S$ *produces a satisfying assignment of* φ *and for any polynomial* p *the set*

$$\{\varphi \in S \mid \mathrm{time}_{M'}(\varphi) > p(|\varphi|)\}$$

is not sparse.

3. *There is a* P*-subset* S *of* SAT*, a non-sparse subset* L *of* S*, and a super-polynomial function* f *such that for any deterministic Turing machine* M' *that on input* $\varphi \in S$ *produces a satisfying assignment of* φ

$$\mathrm{time}_{M'}(\varphi) > f(|\varphi|)$$

for almost every $\varphi \in L$*.*

4. *There is a machine* M *accepting* SAT*, a non-sparse subset* S *of* SAT*, and a super-polynomial function* f *such that for any deterministic Turing machine* M' *that on input* $\varphi \in$ SAT *produces a satisfying assignment of* φ

$$\mathrm{time}_{M'}(\varphi) > f(|\varphi| + \mathrm{time}_M(\varphi))$$

for almost every $\varphi \in S$*.*

Proof. Let pad: $\Sigma^* \times \Sigma^* \to \Sigma^*$ be an injective function that is FP-invertible with the following further properties

1. $\mathrm{pad}(\varphi, w) \in \mathrm{SAT}$ if and only if $\varphi \in \mathrm{SAT}$.
2. from a *sat*-proof for $\psi = \mathrm{pad}(\varphi, w)$ one can easily compute a satisfying assignment for φ.
3. $|\mathrm{pad}(\varphi, w)| = |\varphi| + c(|w| + 1)$ for some constant c.

Notice that there is such a padding function.

To see that item 1 implies item 2, assume that the contrary of 2 holds. We will see that this implies Statement 4 in Theorem 16 which completes the proof of this implication. Let S be an arbitrary P-subset of SAT and let

$$T \;=\; \{\mathrm{pad}(\varphi, w) \mid w \in \Sigma^*, \varphi \in S\}\,.$$

As pad is invertible we also have $T \in \mathsf{P}$. Assuming the contrary of 2 there is a polynomial p and a deterministic Turing machine M' that on input $\varphi \in T$ produces a satisfying assignment of φ such that the set

$$Q \;=\; \{\varphi \in T \mid \mathrm{time}_{M'}(\varphi) > p(|\varphi|)\}$$

is sparse. Let q be a polynomial bound for the density of Q (i. e., $\|Q \cap \Sigma^n\| \leq q(n)$ for each n).

Now a function $g \in \mathsf{FP}$ that on input $\varphi \in S$ yields a satisfying assignment for φ can be computed as follows. Let $q'(n)$ denote the polynomial $q(n+c(n+1))+1$. Let $\varphi \in S$ be an input of length n. First assume that n is sufficient large, i.e., $q'(n) \leq 2^n$. Let $w_1, \ldots, w_{q'(n)}$ denote the $q'(n)$ lexicographically first strings of Σ^n. Now simulate 'in parallel' the computations of M' on input of $\mathrm{pad}(\varphi, w_1)$,

$\ldots$, $\mathrm{pad}(\varphi, w_{q'(n)})$, i. e. perform a simulation in stages, in stage i for each $1 \leq j \leq q'(n)$ simulate the ith step of the computation of M' on $\mathrm{pad}(\varphi, w_j)$; notice that each stage i can be completed in time $(q'(n) \cdot i)^2$. Stop the simulation as soon as the first of those computations produces a satisfying assignment, say e. g. for $\mathrm{pad}(\varphi, w_j)$. From this assignment obtain a satisfying assignment for φ. For the finitely many input lengths n that do not satisfy $q'(n) \leq 2^n$ determine a satisfying assignment by a table look-up.

This construction guarantees that for some polynomial r and every $\varphi \in S$ we obtain a satisfying assignment of φ after at most

$$r(n + \min\{\mathrm{time}_{M'}(\mathrm{pad}(\varphi, w_j)) \mid 1 \leq j \leq q'(n)\})$$

steps. Observe further that the set $\{\mathrm{pad}(\varphi, w_j) \mid 1 \leq j \leq q'(n)\} \subseteq \Sigma^{n+c(n+1)}$ is of cardinality $q'(n) = q(n + c(n+1)) + 1$ and thus cannot be fully contained in Q. Thus by assumption for some w_j M' produces a satisfying assignment of $\mathrm{pad}(\varphi, w_j)$ in time $p(|\mathrm{pad}(\varphi, w_j)|)$. This shows that the time needed for the above computation is bounded by a polynomial.

For the implication $2 \Rightarrow 3$ let S be a set such that 2 is fulfilled. A suitable subset $L \subseteq S$ is obtained by the following construction. Let $M'_1, M'_2, \ldots$ be a (non effective) enumeration of the deterministic Turing machines that on input $\varphi \in S$ produce a satisfying assignment of φ.

Set $n_0 = 0$, and for $k > 0$ let n_k be the smallest number $n > n_{k-1}$ such that the set

$$S_k^n := \{\varphi \in S \cap \Sigma^n \mid \mathrm{time}_{M'_i}(\varphi) > n^k + k \text{ for all } i \leq k\}$$

has cardinality greater than $n^k + k$.

Observe that for each k there is such a number n_k. If this were not the case for some k, then for the least such k the set

$$T = \{\varphi \in S \mid \mathrm{time}_{M'_i}(\varphi) > |\varphi|^k + k \text{ for all } i \leq k\}$$

would be sparse (namely for each $n > n_{k-1}$ we had $\|T \cap \Sigma^n\| \leq n^k + k$). However, this contradicts 2 if we let M' be a deterministic Turing machine that in parallel simulates the machines $M'_1, \ldots, M'_k$ (notice that this parallel simulation is even possible without overhead since k is constant).

Now let $L = \bigcup_{k>0} S_k^{n_k}$, and let $f(n) = n^{k(n)} + k(n)$ where $k(n) = \max\{k \mid n_k \leq n\}$.

Clearly f is super-polynomial, and L is non-sparse. To see that f and L also fulfill the last condition in 3 let M'_i be a deterministic Turing machine that on input $\varphi \in S$ produces a satisfying assignment of φ. By the construction, we have for each $k \geq i$, and for each $\varphi \in S_k^{n_k}$

$$\mathrm{time}_{M'_i}(\varphi) > |\varphi|^k + k = f(|\varphi|).$$

Thus for each $\varphi \in L$ with $|\varphi| \geq n_i$, $\mathrm{time}_{M'_i}(\varphi) > f(|\varphi|)$.

For the proof of the implication $3 \Rightarrow 4$, we observe that item 3 is the negation of item 4 from Theorem 16, whereas item 4 is the negation of item 2 in Theorem 19. Therefore, 3 implies 4 by an argument similar to the one given in the proof of implication $2 \Rightarrow 1$ in Theorem 19. Finally, the implication $4 \Rightarrow 1$ follows directly from Theorem 19. □

6 On the Existence of P-optimal Proof Systems

In Theorem 18 it is observed that *sat* is p-optimal if and only if every optimal proof system is p-optimal. Although the assumption of the mere existence of a p-optimal proof system for SAT is presumably weaker than the assumption that *sat* is p-optimal, it is still equivalent to a quite similar statement, namely that any set with an optimal proof system has a p-optimal proof system. For the proof of this result we use the following observation from [28].

Lemma 21 ([28]). *If L has a (p-)optimal proof system, and $T \leq_m^p L$, then T has a (p-)optimal proof system.*

Using this lemma we can prove the following 'existentially quantified' version of Theorem 18 from the previous section.

Theorem 22. *The following statements are equivalent.*

1. SAT *has a p-optimal proof system.*
2. *Any language L that has an optimal proof system also has a p-optimal proof system.*

Proof. Clearly, item 2 implies item 1, as SAT has an optimal proof system. To see the converse implication assume that SAT has a p-optimal proof system, and let L be an arbitrary nonempty language with an optimal proof system. Let T_L (cf. [28]) be the following language consisting of tuples $\langle M, x, 0^s\rangle$ where M is a deterministic Turing transducer, $s \geq 0$, and $x \in \Sigma^*$.

$$T_L = \{\langle M, x, 0^s\rangle \mid \mathrm{time}_M(x) > s \text{ or } M(x) \in L\}.$$

Notice that T_L is many-one reducible to L. Hence, the assumption that there is an optimal proof system for L implies that T_L has an optimal proof system, say h. Let

$$S_h = \{\langle\langle M, x, 0^s\rangle, 0^l\rangle \mid h \vdash_{\leq l} \langle M, x, 0^s\rangle\}.$$

Clearly $S_h \in \mathsf{NP}$. Therefore by assumption there is a p-optimal proof system g for S_h. Let now f be the following proof system.

$$f(w) = \begin{cases} y & \text{if } g(w) = \langle\langle M, x, 0^s\rangle, 0^l\rangle, \\ & \quad \text{and on input } x,\ M \text{ outputs } y \text{ in } \leq s \text{ steps}, \\ \text{undef.} & \text{otherwise.} \end{cases}$$

First notice that $y \in L$ if $f(w) = y$. This is due to the fact that $g(w) = \langle\langle M, x, 0^s\rangle, 0^l\rangle$ implies $\langle\langle M, x, 0^s\rangle, 0^l\rangle \in S_h$ which in turn implies $\langle M, x, 0^s\rangle \in T_L$. On the other hand, each $y \in L$ is easily seen to have an f-proof, hence f is a proof system for L.

We now show that f p-simulates every proof system f' for L. Assume that f' is computed by the transducer $M_{f'}$ in polynomial time $p(n)$. Observe that $\langle M_{f'}, x, 0^{p(|x|)}\rangle \in T_L$ for any $x \in \Sigma^*$. Hence, one may define a proof system for T_L such that for any x the tuple $\langle M_{f'}, x, 0^{p(|x|)}\rangle$ has the short proof $1x$. Consequently, due to the optimality of h, there is a polynomial q such that

$\langle M_{f'}, x, 0^{p(|x|)}\rangle$ has an h-proof of size $\leq q(|x|)$. Now $\langle\langle M_{f'}, x, 0^{p(|x|)}\rangle, 0^{q(|x|)}\rangle \in S_h$ for any x, and one may define a proof system g' for S_h with $g'(1x) = \langle\langle M_{f'}, x, 0^{p(|x|)}\rangle, 0^{q(|x|)}\rangle$ for any x. As g is p-optimal, g p-simulates g', i.e., there is a function $t \in \mathsf{FP}$ such that $g(t(1x)) = g'(1x) = \langle\langle M_{f'}, x, 0^{p(|x|)}\rangle, 0^{q(|x|)}\rangle$. Observe now that $f(t(1x)) = f'(x)$ for any x. Hence f p-simulates f'. □

As shown in [28], the assumption that SAT and TAUT both have p-optimal proof systems implies that $\mathsf{NP} \cap \mathsf{coNP}$ has a many-one complete set. In fact, due to Theorem 22 it suffices to assume that SAT has a p-optimal proof system and TAUT only has an optimal proof system. Together with Proposition 3 we obtain:

Corollary 23. *If* SAT *has a p-optimal and* TAUT *has an optimal proof system, then* $\mathsf{NP} \cap \mathsf{coNP}$ *has a many-one complete set, and* NPSV_t *has a many-one complete function.*

Next we show that a p-optimal proof system for SAT implies a complete function for the class NPMV_t. The proof uses ideas from [28].

Theorem 24. *If* SAT *has a p-optimal proof system, then* NPMV_t *has a many-one complete function.*

Proof. Consider the NP-set $L = \{\langle N, x, 0^s\rangle \mid$ there is an accepting path of N on input x of length $\leq s\}$. If SAT has a p-optimal proof system, then due to Lemma 21 there is a p-optimal proof system h for L. We show that the following function g is complete for NPMV_t.

> If the input is a tuple $u = \langle N, x, 0^s, w\rangle$ with the property that $h(w) = \langle N, x, 0^s\rangle$, then set-$g(u) = \{y \mid y$ is an output of N on an accepting path of length $\leq s$ on input $x\}$. Otherwise set-$g(u) = \{\lambda\}$.

It is clear that g is in NPMV_t. To see that g is hard for NPMV_t let $f \in \mathsf{NPMV}_t$ be computed by a nondeterministic Turing machine N_f with a polynomial time-bound p. It is easy to see that there is a proof system h' for L with $h'(1x) = \langle N_f, x, 0^{p(|x|)}\rangle$ for any x. As h p-simulates h' there is a function $t \in \mathsf{FP}$ such that $h(t(1x)) = \langle N_f, x, 0^{p(|x|)}\rangle$ for any x. So, $x \mapsto \langle N_f, x, 0^{p(|x|)}, t(1x)\rangle$ is a many-one reduction from f to g. □

By Theorem 9 we obtain:

Corollary 25. *If* SAT *has a p-optimal proof system, then there exists a strongly many-one complete disjoint* coNP*-pair.*

In the following table we collect some of the implications proven in this section.

Assumption	Consequence
p-optimal proof system for SAT and optimal proof system for TAUT	complete set for $\mathsf{NP} \cap \mathsf{coNP}$ and complete function for NPSV_t
optimal proof system for TAUT	complete function for NPkV, $k \geq 1$
p-optimal proof system for SAT	complete function for NPMV_t and complete disjoint coNP-pair

7 Conclusion

We have shown that the assumption that certain proof systems are (p-)optimal can be used to derive collapse results. Also we presented some relations between completeness assumptions for different classes. It would be interesting to know whether these observations can be extended to further proof systems and promise classes.

References

1. N. Alon and R. B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.
2. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin Heidelberg, second edition, 1995.
3. O. Beyersdorff. Classes of representable disjoint NP-pairs. *Theoretical Computer Science*, 377(1–3):93–109, 2007.
4. O. Beyersdorff. The deduction theorem for strong propositional proof systems. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4855 of *Lecture Notes in Computer Science*, pages 241–252. Springer-Verlag, Berlin Heidelberg, 2007.
5. O. Beyersdorff. Tuples of disjoint NP-sets. *Theory of Computing Systems*, 43(2):118–135, 2008.
6. M. L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, and T. Pitassi. Non-automatizability of bounded-depth Frege proofs. *Computational Complexity*, 13(1–2):47–68, 2004.
7. M. L. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997.
8. M. L. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.
9. R. Book, T. Long, and A. L. Selman. Quantitative relativizations of complexity classes. *SIAM Journal on Computing*, 13(3):461–487, 1984.
10. H. Buhrman, L. Fortnow, M. Koucký, J. D. Rogers, and N. K. Vereshchagin. Inverting onto functions and polynomial hierarchy. In *Proc. 2nd International Computer Science Symposium in Russia*, volume 4649 of *Lecture Notes in Computer Science*, pages 92–103. Springer-Verlag, Berlin Heidelberg, 2007.
11. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.
12. W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, 22(3):269–285, 1957.
13. P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. *SIAM Journal on Computing*, 28(5):1759–1782, 1999.
14. S. Fenner, L. Fortnow, A. Naik, and J. Rogers. Inverting onto functions. *Information and Computation*, 186(1):90–103, 2003.
15. S. Fenner, S. Homer, M. Ogihara, and A. L. Selman. Using oracles that compute values. *SIAM Journal on Computing*, 26(4):1043–1065, 1997.
16. L. Fortnow and J. Rogers. Separability and one-way functions. *Computational Complexity*, 11(3–4):137–157, 2002.
17. C. Glaßer, A. L. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. *Information and Computation*, 200(2):247–267, 2005.
18. C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.
19. 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.
20. 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.

21. C. Glaßer, A. L. Selman, and L. Zhang. The informational content of canonical disjoint NP-pairs. In *Proc. 13th International Computing and Combinatorics Conference*, volume 4598 of *Lecture Notes in Computer Science*, pages 307–317. Springer-Verlag, Berlin Heidelberg, 2007.
22. J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.
23. A. Große and H. Hempel. On functions and relations. In *Proc. 4th International Conference on Discrete Mathematics and Theoretical Computer Science*, volume 2731 of *Lecture Notes in Computer Science*, pages 181–192. Springer-Verlag, Berlin Heidelberg, 2003.
24. L. Hemaspaandra, A. Hoene, A. Naik, M. Ogihara, A. L. Selman, T. Thierauf, and J. Wang. Nondeterministically selective sets. *International Journal of Foundations of Computer Science*, 6(4):403–416, 1995.
25. R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proc. 3rd Structure in Complexity Theory Conference*, pages 29–38, 1988.
26. K.-I. Ko. On some natural complete operators. *Theoretical Computer Science*, 37:1–30, 1985.
27. J. Köbler and J. Messner. Is the standard proof system for SAT P-optimal? In *Proc. 20th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 1974 of *Lecture Notes in Computer Science*, pages 361–372. Springer-Verlag, Berlin Heidelberg, 2000.
28. J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.
29. S. Kosub, H. Schmitz, and H. Vollmer. Uniform characterizations of complexity classes of functions. *Int. J. Found. Comput. Sci.*, 11(4):525–551, 2000.
30. J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.
31. J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.
32. 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.
33. J. Krajíček and P. Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.
34. M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.
35. T. J. Long. On γ-reducibility versus polynomial time many-one reducibility. *Theoretical Computer Science*, 14:91–101, 1981.
36. J. Messner. On optimal algorithms and optimal proof systems. In *Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 541–550. Springer-Verlag, Berlin Heidelberg, 1999.
37. D. Mundici. Tautologies with a unique Craig interpolant, uniform vs. nonuniform complexity. *Annals of Pure and Applied Logic*, 27:265–273, 1984.
38. A. Naik, J. Rogers, J. Royer, and A. L. Selman. A hierarchy based on output multiplicity. *Theoretical Computer Science*, 207(1):131–157, 1998.
39. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
40. P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.
41. P. Pudlák. The lengths of proofs. In S. R. Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, Amsterdam, 1998.
42. P. Pudlák. On reducibility and symmetry of disjoint NP-pairs. *Theoretical Computer Science*, 295(1–3):323–339, 2003.
43. P. Pudlák and J. Sgall. Algebraic models of computation and interpolation for algebraic proof systems. In P. W. Beame and S. R. Buss, editors, *Proof Complexity and Feasible Arithmetic*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 279–296. American Mathematical Society, 1998.
44. A. A. Razborov. Lower bounds on the monotone complexity of boolean functions. *Doklady Akademii Nauk SSSR*, 282:1033–1037, 1985. English translation in: Soviet Math. Doklady, 31, pp. 354–357.

45. A. A. Razborov. On provably disjoint NP-pairs. Technical Report TR94-006, Electronic Colloquium on Computational Complexity, 1994.
46. Z. Sadowski. On an optimal quantified propositional proof system and a complete language for NP ∩ 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.
47. Z. Sadowski. On an optimal deterministic algorithm for SAT. In *Proc. 12th Workshop on Computer Science Logic*, volume 1584 of *Lecture Notes in Computer Science*, pages 179–187. Springer-Verlag, Berlin Heidelberg, 1998.
48. Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets of TAUT. *Theoretical Computer Science*, 288(1):181–193, 2002.
49. Z. Sadowski. Optimal proof systems, optimal acceptors and recursive presentability. *Fundamenta Informaticae*, 79(1–2):169–185, 2007.
50. F. Scarcello and D. Saccà. Multivalued functions: Turing characterizations and complexity results. Unpublished Manuscript, 2002.
51. U. Schöning. *Complexity and Structure*, volume 211 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 1986.
52. U. Schöning and J. Torán. A note on the size of Craig interpolants. In T. Schwentick, D. Thérien, and H. Vollmer, editors, *Circuits, Logic, and Games*, number 06451 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007.
53. A. L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.
54. A. L. Selman. Much ado about functions. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, pages 198–212, 1996.
55. L. Valiant. Relative complexity of checking and evaluation. *Information Processing Letters*, 5(1):20–23, 1976.
56. V. Zankó. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2(1):77–82, 1991.