Proceedings Paper:
Beyersdorff, O (2009) On the existence of complete disjoint NP-pairs. In: SYNASC 2009 - 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. 11th International Symposium on Symbolic & Numeric Algorithms for Scientific Computing, 26-29 Sep 2009, Timisoara, Romania. IEEE, 282 - 289 . ISBN: 978-1-4244-5910-0.

https://doi.org/10.1109/SYNASC.2009.9

On the Existence of Complete Disjoint NP-Pairs

Olaf Beyersdorff
Institut für Theoretische Informatik, Leibniz-Universität Hannover
Appelstraße 4, 30167 Hannover, Germany
Email: beyersdorff@thi.uni-hannover.de

***Abstract*—Disjoint NP-pairs are an interesting model of computation with important applications in cryptography and proof complexity. The question whether there exists a complete disjoint NP-pair was posed by Razborov in 1994 and is one of the most important problems in the field. In this paper we prove that there exists a complete disjoint NP-pair which is computed with access to a very weak oracle (a tally NP-oracle).**

In addition, we exhibit candidates for complete NP-pairs and apply our results to a recent line of research on the construction of hard tautologies from pseudorandom generators.

I. Introduction

Disjoint NP-pairs have been introduced by Grollmann and Selman [1] as a complexity-theoretic tool to model the security of public-key cryptosystems. Subsequently, Razborov [2] established the link between disjoint NP-pairs and propositional proof complexity by associating a canonical pair to a proof system. This connection was further developed by Pudlák [3] and Krajíček [4], [5] and is by now a fruitful field of research contributing both to structural and proof complexity (cf. [6]–[11] and [12] for a survey). Due to these applications there has been strong recent interest in the theory of disjoint NP-pairs and new exciting results have been obtained which deepened our understanding of these objects [6], [7], [13]–[15].

One of the most prominent questions in the field, posed by Razborov [2], asks whether there exist complete disjoint NP-pairs. While this problem has been open for 15 years, some partial answers are known. First, the existence of optimal propositional proof systems yields a sufficient condition for the existence of complete NP-pairs [2], [16]. Second, the question was shown to be largely independent of the underlying reduction, namely, complete pairs exist under strong many-one reductions if and only if they exist under a rather weak variant of Turing reductions [14]. Third, the question was studied in the relativised setting and was shown to receive positive and negative answers under suitable oracles [13], [14].

Our contribution here is to show that there exists a complete disjoint NP-pair under a very weak oracle. More precisely, we show that there is a pair (C_1, C_2) where the components C_i are computed in nondeterministic polynomial time with access to a tally NP-oracle such that every disjoint NP-pair strongly many-one reduces to (C_1, C_2). We remark that this result is considerably different from the oracle results in [13] where the oracles are very complex.

Our completeness result connects to a recent line of research, initiated by Cook and Krajíček [17], that determines the power of proof systems computable in polynomial time with the help of advice. In particular, Cook and Krajíček proved that there exists an optimal propositional proof system with only one bit of advice. In [18] we have shown that instead of using a small amount of advice it also suffices to use a sparse NP-oracle. Thus, in the same spirit as in [16], our present result transfers the optimality result on proof systems to a completeness result for promise classes. We state a general theorem which applies to a large class of promise classes with promise conditions in coNP.

In the second part of the paper, we apply our completeness results from the first part to a recent research agenda aiming at the construction of hard formulas for propositional proof systems from pseudorandom generators (called proof complexity generators). The theory of proof complexity generators was developed by Krajíček [4], [19]–[21] and Alekhnovich, Ben-Sasson, Razborov, and Wigderson [22], [23]. It aims at proving lower bounds to the proof size of strong proof systems like Frege systems and their extensions which constitutes a major challenge in propositional proof complexity. So far this program has proved to be successful for weak systems like resolution [4], [22]. Here we give a characterization of the hardness of these formulas for strong proof systems in terms of disjoint NP-pairs. Whether such a characterization helps to solve the original problem remains open. But it provides further evidence that disjoint NP-pairs are applicable to interesting, seemingly unconnected areas.

The paper is organized as follows. After reviewing basic notions from the theory of proof systems and NP-pairs in Section II, we prove in Section III the existence of a complete disjoint NP-pair under a tally NP-oracle. We also generalize the result to further promise classes and derive sufficient conditions for the existence of complete NP-pairs without oracle access. In Section IV we exhibit viable candidates for such complete NP-pairs arising from strong propositional proof systems. Finally, Section V discusses the application of these results to the theory of proof complexity generators.

II. Preliminaries

We assume basic familiarity with complexity classes (cf. [24]). Throughout the paper we fix the alphabet $\Sigma = \{0,1\}$. A set $A \subseteq \Sigma^*$ is *sparse* if there exists a polynomial p such that for each $n \in \mathbb{N}$, $|A \cap \Sigma^n| \leq p(n)$. A sparse set A is called *tally* if $A \subseteq \{1^n \mid n \in \mathbb{N}\}$. The set of all sparse and tally sets are denoted by Sparse and Tally, respectively.

A. Propositional Proof Systems

Propositional proof systems were defined in a very general way by Cook and Reckhow [25] as polynomial-time computable functions P which have as its range the set of all tautologies. A string π with $P(\pi) = \varphi$ is called a P-proof of the tautology φ. By $P \vdash_{\leq m} \varphi$ we indicate that there is a P-proof of φ of length $\leq m$. If Φ is a set of propositional formulas we write $P \vdash_* \Phi$ if there is a polynomial p such that $P \vdash_{\leq p(|\varphi|)} \varphi$ for all $\varphi \in \Phi$. If $\Phi = \{\varphi_n \mid n \geq 0\}$ is a sequence of formulas we also write $P \vdash_* \varphi_n$ instead of $P \vdash_* \Phi$.

Proof systems are compared according to their strength by simulations introduced in [25] and [26]. Given two proof systems P and S we say that S *simulates* P (denoted by $P \leq S$) if there exists a polynomial p such that for all tautologies φ and P-proofs π of φ there is an S-proof π' of φ with $|\pi'| \leq p(|\pi|)$. A proof system is called *optimal* if it simulates all proof systems. Whether or not optimal proof systems exist is an open problem posed by Krajíček and Pudlák [26].

B. Disjoint NP-Pairs

A pair (A,B) is called a *disjoint* NP*-pair* if $A, B \in \mathsf{NP}$ and $A \cap B = \emptyset$. Grollmann and Selman [1] defined the following reduction between disjoint NP-pairs (A,B) and (C,D): $(A,B) \leq_p (C,D)$ if there exists a polynomial time computable function f such that $f(A) \subseteq C$ and $f(B) \subseteq D$. If f performs a $\leq_p$-reduction from (A,B) to (C,D), then f is also allowed to map elements from the complement of $A \cup B$ to C or D. Therefore $f : (A,B) \leq_p (C,D)$ does not imply in general that f is a many-one reduction between A and C or between B and D. This, however, is the case for the following stronger reduction:

Definition 1 (Köbler, Messner, Torán [16])**.** *A disjoint* NP*-pair* (A,B) *is* strongly reducible *to a disjoint* NP*-pair* (C,D)*, denoted by* $(A,B) \leq_s (C,D)$*, if there exists a polynomial time computable function* f *such that* $f^{-1}(C) = A$ *and* $f^{-1}(D) = B$*.*

Equivalently, we can view $\leq_s$ as a reduction between triples. In addition to the two conditions $f(A) \subseteq C$ and $f(B) \subseteq D$ for $\leq_p$ we also require $f(\overline{A \cup B}) \subseteq \overline{C \cup D}$.

The reduction $\leq_s$ now has the property that if f realizes a $\leq_s$-reduction from (A,B) to (C,D), then f is simultaneously a many-one-reduction between A and C as well as between B and D. Clearly, this also serves as a characterization of $\leq_s$, namely:

Proposition 2. *Let* (A,B) *and* (C,D) *be disjoint* NP*-pairs. Then* $(A,B) \leq_s (C,D)$ *if and only if there exists a function* $f \in \mathsf{FP}$ *such that* $f : A \leq_m^p C$ *and* $f : B \leq_m^p D$*.*

Obviously $\leq_s$ is a refinement of $\leq_p$. It is indeed a proper refinement as shown by Glaßer, Selman, and Sengupta [14].

III. Complete Disjoint NP-Pairs under a Tally NP-Oracle

Here we prove our main result stating that a very weak oracle suffices to obtain a complete disjoint NP-pair.

Theorem 3. *There exists a strongly many-one complete disjoint* NP*-pair under a tally* NP*-oracle, i.e., there exists a tally set* $A \in \mathsf{NP}$ *and a disjoint pair* (C_1, C_2) *such that the following holds:*

1) *the components* C_1 *and* C_2 *are computable in* NP^A *with only one query to the oracle* A*, and*
2) *every disjoint* NP*-pair strongly many-one reduces to* (C_1, C_2)*.*

Proof: We choose a polynomial-time computable and invertible tupling function $\langle\cdot\rangle$ on Σ^* which is injective on lengths, i.e., for all strings $x_1, \ldots, x_n, y_1, \ldots, y_n \in \Sigma^*$, $|\langle x_1, \ldots, x_n\rangle| = |\langle y_1, \ldots, y_n\rangle|$ implies $|x_i| = |y_i|$ for $i = 1, \ldots, n$. We also choose a polynomial-time computable encoding of nondeterministic Turing machines by natural numbers. In the following we do not distinguish in notation between a machine and its encoding. If we represent a natural number n in unary, we write it as 1^n.

We define the oracle set A as follows:

$$\begin{aligned} A = \{1^m \mid\ & m = |\langle 1^M, 1^N, 1^n, 1^t\rangle| \text{ where} \\ & M, N, n, t \in \mathbb{N};\ M, N \text{ encode} \\ & \text{nondeterministic Turing machines and} \\ & \text{there exists some } x \in \Sigma^n \text{ such that} \\ & \text{both } M \text{ and } N \text{ accept } x \text{ in time} \leq t\ \}\ . \end{aligned}$$

Intuitively, the set A collects all pairs of nondeterministic machines which accept a common element. Hence, if $1^{|\langle 1^M, 1^N, 1^n, 1^t\rangle|} \in A$, then M, N will not define a disjoint NP-pair.

By definition, the set A is tally. Let us verify that $A \in \mathsf{NP}$. Because of the length injectivity of the tupling function, a number $m \in \mathbb{N}$ already uniquely determines the tuple $\langle 1^M, 1^N, 1^n, 1^t\rangle$ with $|\langle 1^M, 1^N, 1^n, 1^t\rangle| = m$. Therefore, on input 1^m we can first determine the entries M, N, n, t and then verify that M, N indeed encode nondeterministic Turing machines. Next we guess a string $x \in \Sigma^n$ and nondeterministically simulate both M and N on input x for at most t steps. If both computations accept, then we accept the input 1^m, otherwise we reject.

Now we define the disjoint pair (C_1, C_2) which will be $\leq_s$-hard for all disjoint NP-pairs. The component C_1 takes elements of the form

$$\langle 1^M, 1^N, x, 1^t \rangle$$

with natural numbers M, N, t and a string $x \in \Sigma^*$. On such input, C_1 first queries the string $\langle 1^M, 1^N, 1^{|x|}, 1^t \rangle$ to the oracle A. If the answer is negative, then we simulate M on input x for at most t steps and answer according to the output of this simulation. If the answer is positive or if the simulation does not terminate in t steps, then we reject. The component C_2 is defined analogously, except that we use the machine N instead of M for the simulation.

To verify the hardness of (C_1, C_2), let (B_1, B_2) be a disjoint NP-pair. Let M, N be nondeterministic Turing machines for B_1, B_2, respectively, and let p be a polynomial bounding the running time of both M and N. Then (B_1, B_2) strongly many-one reduces to (C_1, C_2) via the reduction

$$x \mapsto \langle 1^M, 1^N, x, 1^{p(|x|)} \rangle \ .$$

The correctness of the reduction is easy to verify. ∎

It is known that there is a close connection between disjoint NP-pairs and functions from NPSV, single-valued functions computable in nondeterministic polynomial time (cf. [13], [27], [28] for definitions and background information). Using this correspondence we can formulate Theorem 3 differently as:

Corollary 4. *There exists a tally* NP*-set A and a function $f \in \mathsf{NPSV}^A$ such that every function from* NPSV *is many-one reducible to f.*

From Theorem 3 we also get a sufficient condition for the existence of complete disjoint NP-pairs:

Corollary 5. *If $\mathsf{NP} = \mathsf{NP}^{\mathsf{NP} \cap \mathsf{Tally}}$, then there exist $\leq_s$-complete disjoint* NP*-pairs.*

We can rephrase this corollary using the notion of low sets from [29]. Recall that a set $A \in \mathsf{NP}$ is *low* for the nth level Σ_n^p of the polynomial hierarchy if $(\Sigma_n^p)^A \subseteq \Sigma_n^p$. Intuitively, if a set A is low for Σ_n^p, then A is useless as an oracle for the class Σ_n^p. All sets $A \in \mathsf{NP}$ which are low for Σ_n^p are collected in the nth level L_n of the low hierarchy. Using this terminology, we can express Corollary 5 differently as:

Corollary 6. *If $\mathsf{NP} \cap \mathsf{Tally} \subseteq \mathsf{L}_1$, then there exist $\leq_s$-complete disjoint* NP*-pairs.*

Whether or not $\mathsf{NP} \cap \mathsf{Tally} \subseteq \mathsf{L}_1$ is open, but Ko and Schöning [30] have shown that $\mathsf{NP} \cap \mathsf{Sparse} \subseteq \mathsf{L}_2$.

We remark that Theorem 3 allows for a generalization to other promise classes. In order to state the result, let us review the general notion of a promise class as defined e.g. in [16]. A promise R is described as a binary predicate between nondeterministic polynomial-time Turing machines N and strings x, i.e., $R(N, x)$ means that N obeys promise R on input x. A machine N is called an *R-machine* if N obeys R on any input $x \in \Sigma^*$. Given a promise predicate R, we define the language class

$$\mathsf{C}_R = \{ L(N) \mid N \text{ is an R-machine} \}$$

and call it the promise class generated by R.

An important question is how hard it is to verify the promise for a given instance. In particular, we are interested in promise classes with promise conditions in coNP. This notion is made precise in the following definition:

Definition 7. *A promise condition R is a* coNP-promise *if there exist a language $L \in \mathsf{coNP}$ and a polynomial-time computable function $corr : \Sigma^* \times \Sigma^* \times 1^* \to \Sigma^*$ such that the following conditions hold:*

1) Correctness: *For every polynomial-time clocked Turing machine N, for every $x \in \Sigma^*$ and $m \in \mathbb{N}$, if $corr(x, N, 1^m) \in L$, then N obeys promise R on input x.*
2) Completeness: *For every R-machine N with polynomial time bound p, the set*

$$Correct(N) = \{ corr(x, N, 1^{p(|x|)}) \mid x \in \Sigma^* \}$$

is a subset of L.
3) Local recognizability: *For every Turing machine N, the set $Correct(N)$ is polynomial-time decidable.*

Usually, promise classes possess a *universal machine*, i.e., there exists a universal machine U_R which, given an R-machine N, input x, and time bound 1^m, efficiently simulates $N(x)$ for m steps such that U_R obeys promise R on $\langle N, x, 1^m \rangle$.

For promise classes with universal machines and promise conditions in coNP we can state the following general result:

Theorem 8. *Let C be a promise language (or function) class defined via a* coNP*-promise. Let C have a universal machine. Then exists a tally* NP*-oracle A such that C^A contains a language (or function) which is many-one hard for C.*

Proof: The proof proceeds similarly as the proof of Theorem 3. We just indicate the necessary changes. The oracle set A now contains all machines which violate the promise condition on some given length, i.e.,

$$\begin{aligned} A = \{ \langle 1^N, 1^n, 1^t \rangle \mid N, n, t \in \mathbb{N}, \\ N \text{ is a nondeterministic Turing machine,} \\ \text{and there exists some } x \in \Sigma^n \text{ such that} \\ corr(x, N, 1^t) \notin L \} \ , \end{aligned}$$

where L is the coNP-set from Definition 7 in which the promise of C is expressible. As $L \in \mathsf{coNP}$, the set A is a tally NP-set. The hard set for C will now contain elements $\langle 1^N, x, 1^t \rangle$ where N is a correct C-machine on input length $|x|$ (this is verified via the oracle), and N accepts x in time $\leq t$ (here we need the universal machine for C). ∎

IV. Canonical Candidates for Complete Pairs

In this section we approach the question whether complete disjoint NP-pairs exist without making further assumptions. We will first comment on the general difficulty to construct a complete pair and then present natural candidates for strongly many-one complete disjoint NP-pairs.

Complexity classes are usually defined by a machine model to which resource bounds are imposed. A complexity class is *syntactic* if the machines can be appropriately standardized such that there exists an easy test which verifies that all these standardized machines define indeed languages from the complexity class (cf. [31]). For syntactic classes there is a canonical way how to define complete languages. Namely, if $\mathcal{M}$ denotes the set of all standardized machines with implicit resource bounds, then

$$\{(M, x) \mid M \in \mathcal{M} \text{ and } M(x) \text{ accepts}\}$$

is complete for the respective complexity class. For example the syntactic class NP has the following canonical $\leq_m^p$-complete language

$$\{(M, x, 1^m) \mid M \text{ is a nondeterministic Turing machine that accepts } x \text{ in } \leq m \text{ steps}\} \ .$$

The machine model for disjoint NP-pairs consists of pairs of nondeterministic polynomial-time bounded Turing machines that do not accept any element in common. This, however, is not a syntactic definition as we cannot test whether two given nondeterministic Turing machines indeed accept disjoint languages. In fact, by the theorem of Rice [32] the set

$$\{(M_1, M_2) \mid M_1 \text{ and } M_2 \text{ are nondeterministic Turing machines such that } L(M_1) \cap L(M_2) = \emptyset\}$$

is undecidable. Therefore, constructing complete disjoint NP-pairs via the above method fails.

If we restrict the class of all pairs to those disjoint NP-pairs whose disjointness is shortly provable in some fixed proof system P, then the situation is different. The machine model now consists of pairs (M_1, M_2) of polynomial-time nondeterministic Turing machines such that the disjointness of $L(M_1)$ and $L(M_2)$ has polynomial-size P-proofs for suitable propositional descriptions of M_1 and M_2. These propositional descriptions lead to the function $corr(x, M_1, M_2, 1^n)$ from Definition 7 which is computable in polynomial time (details are given in the proof of Theorem 9 below). As further the polynomial-size P-proofs of $Correct(M_1, M_2)$ can be guessed and verified in polynomial time, the process of checking that (M_1, M_2) defines a disjoint NP-pair can be performed in nondeterministic polynomial time. Hence, for a propositional proof system P we can form a syntactic class

$$\mathsf{DNPP}(P) = \{(M_1, M_2) \mid P \vdash_* Correct(M_1, M_2)\} \ .$$

If a pair (A, B) is contained in $\mathsf{DNPP}(P)$ we also say that (A, B) is *representable* in P.

For the syntactic class $\mathsf{DNPP}(P)$ we can define hard languages in the canonical way. Translating this canonical hard language to the propositional level we arrive at a pair $W(P) = (W_1(P), W_2(P))$ with

$$\begin{aligned} W_1(P) = \{ \langle \varphi(\bar{x}, \bar{y}), \psi(\bar{x}, \bar{z}), a, 1^m \rangle \mid \\ \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \{\bar{x}\},\ \varphi(a, \bar{y}) \in \mathrm{SAT}, \\ \text{and } P \vdash_{\leq m} \neg\varphi(\bar{x}, \bar{y}) \vee \neg\psi(\bar{x}, \bar{z})\} \\ W_2(P) = \{ \langle \varphi(\bar{x}, \bar{y}), \psi(\bar{x}, \bar{z}), a, 1^m \rangle \mid \\ \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \{\bar{x}\},\ \psi(a, \bar{z}) \in \mathrm{SAT}, \\ \text{and } P \vdash_{\leq m} \neg\varphi(\bar{x}, \bar{y}) \vee \neg\psi(\bar{x}, \bar{z})\} \ . \end{aligned}$$

In the components $W_1(P)$ and $W_2(P)$ the propositional formulas $\varphi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ describe the Turing machines M_1 and M_2 for inputs of length $|\bar{x}|$, i.e., for all $a \in \Sigma^{|\bar{x}|}$, M_1 accepts a if and only if $\varphi(a, \bar{y})$ is satisfiable (and similarly for M_2 and ψ). In the formulas φ, ψ, the variables $\bar{x}$ are reserved for the input whereas the variables $\bar{y}$ and $\bar{z}$ take the witness and auxiliary information necessary for the computation of the machines M_1 and M_2. The P-proofs of length $\leq m$ certify the disjointness of $L(M_1)$ and $L(M_2)$. Finally, the satisfiability conditions on $\varphi(a, \bar{y})$ and $\psi(a, \bar{z})$ describe that M_1 and M_2, respectively, accept the input a.

Let us argue that $(W_1(P), W_2(P))$ is indeed a disjoint NP-pair. Clearly, both components are in NP. To verify the disjointness, assume that $\langle \varphi(\bar{x}, \bar{y}), \psi(\bar{x}, \bar{z}), a, 1^m \rangle$ is contained in $W_1(P)$. Since we have a P-proof, the formula $\neg\varphi(\bar{x}, \bar{y}) \vee \neg\psi(\bar{x}, \bar{z})$ is a tautology. By assumption, $\varphi(a, \bar{y})$ is satisfiable and hence $\psi(a, \bar{z})$ must be a tautology. Therefore, $\neg\psi(a, \bar{z})$ is unsatisfiable which implies $\langle \varphi(\bar{x}, \bar{y}), \psi(\bar{x}, \bar{z}), a, 1^m \rangle \notin W_2(P)$.

Our next result states that the pair $W(P)$ is the canonical choice for a $\leq_s$-hard and, for many natural systems, even $\leq_s$-complete pair for $\mathsf{DNPP}(P)$.

Theorem 9.

1) *For any propositional proof system P the pair $W(P)$ is $\leq_s$-hard for the class $\mathsf{DNPP}(P)$.*
2) *Let P be a proof system of the form $EF + \Phi$ with some polynomial-time computable set $\Phi \subseteq \mathrm{TAUT}$. Then the pair $W(P)$ is $\leq_s$-complete for $\mathsf{DNPP}(P)$.*

Proof: For the first item, let P be a proof system and let $(A, B) \in \mathsf{DNPP}(P)$. Let M and N be nondeterministic machines with polynomial running time $p(n)$ which compute the components A and B, respectively. We construct the function $corr$ as

$$corr(x, M, N, 1^{p(|x|)}) = \neg\varphi_{|x|}(\bar{x}, \bar{y}) \vee \neg\psi_{|x|}(\bar{x}, \bar{z})$$

where φ_i and ψ_i are sequences of propositional formulas describing the machines M and N as explained above. As $(A, B) \in \mathsf{DNPP}(P)$ there exists a polynomial q such that

for all $x \in \Sigma^*$

$$P \vdash_{\leq q(|x|)} corr(x, M, N, 1^{p(|x|)}) \ .$$

It is then straightforward to verify that

$$a \mapsto \langle \neg\varphi_{|a|}(\bar{x}, \bar{y}), \neg\psi_{|a|}(\bar{x}, \bar{z}), a, 1^{q(|a|)} \rangle$$

realizes the reduction $(A, B) \leq_s (W_1(P), W_2(P))$.

For the second item it remains to show $W(P) \in \mathsf{DNPP}(P)$ for proof systems P of the form $EF + \Phi$ with a polynomial-time decidable set $\Phi \subseteq \mathrm{TAUT}$. For this we have to construct propositional representations of $W(P)$ such that P admits short proofs for the disjointness of $W_1(P)$ and $W_2(P)$ with respect to these representations. A direct construction of such P-proofs would be quite tedious, but we can use the correspondence of extensions of EF to first-order arithmetic theories (cf. [33], [34] for background information).

In this framework, the argument proceeds as follows: first we choose natural arithmetic formulas defining the components of $W(P)$. We now argue in the arithmetic theory S^1_2 augmented by the reflection principle of P (reflection is a strong way to state the correctness of the proof system P). Using the reflection principle it is then straightforward to verify the disjointness of $W(P)$ with respect to the chosen arithmetic representations by a first-order proof. This proof can be translated into a sequence of polynomial-size propositional proofs in the system P, yielding representability of $W(P)$ in P. For a more detailed description of this procedure we refer to [8], [9]. ■

Let us mention that for strong proof systems, Razborov's canonical pair [2] and Pudlák's interpolation pair [3] are two other candidates for complete disjoint NP-pairs. Moreover, these pairs relate to important properties of proof systems. Namely, the canonical pair captures the reflection principle and is linked to the automatizability of the proof system [3], [35], while the interpolation pair expresses the feasible interpolation property. The advantage of our W-pair is that we can show its $\leq_s$-hardness for $\mathsf{DNPP}(P)$ for every proof system P, whereas to prove such a result for the reflection or interpolation pair requires some additional assumptions on the proof system (cf. [9]).

Whether or not disjoint NP-pairs exist must remain open. However, in the light of results like Theorem 9, the pair $W(P)$ (as well as the canonical or interpolation pair of P) is a good candidate for a complete disjoint NP-pair for strong propositional proof systems P (such as $P = EF$).

V. Pseudorandom Generators in Propositional Proof Complexity

This section is devoted to a potential application of the results of the previous sections for the construction of hard tautologies from pseudorandom generators (called τ-*formulas*). To employ pseudorandom generators as the basis for proving lower bounds to the proof size in propositional proof systems was independently suggested by Krajíček [4], [19], [20] and by Alekhnovich, Ben-Sasson, Razborov and Wigderson [22]. These τ-formulas are candidates for tautologies without polynomially long proofs in strong proof systems like EF and their extensions. Proving super-polynomial lower bounds for strong proof systems constitutes a major open problem in propositional proof complexity. The aim of this section is to illustrate that the hardness of τ-formulas can be expressed by properties of disjoint NP-sets .

We recall some terminology from [4]. Let $C = (C_n)_{n\in\mathbb{N}}$ be a family of polynomial-size Boolean circuits such that C_n is a circuit with n input and $m(n) > n$ output bits with some polynomial m. Functions f computed by such families C are called *polynomially stretching (p-stretching)*.

For $b \in \{0,1\}^{m(n)}$ we consider propositional formulas $\tau(C)_b$. The formula $\tau(C)_b$ has propositional variables $p_1, \ldots, p_n$ for the bits of the input of C_n, $q_1, \ldots, q_{m(n)}$ for the bits of the output of C_n and $r_1, \ldots, r_{n^{O(1)}}$ for the inner nodes of C_n. The formula $\tau(C)_b$ expresses that if $\bar{r}$ are correctly computed according to C_n from the input variables $\bar{p}$, then the values of the output variables $\bar{q}$ are different from the bits of b. The formula $\tau(C)_b$ is a tautology if and only if $b \notin \mathrm{rng}(f)$. But apparently $\tau(C)_b$ does not only depend on $\mathrm{rng}(f)$ but also on the particular circuits C_n used for the computation of f.

The formulas $\tau(C)$ from a circuit family C_n are called *hard* for a proof system P, if there does not exist a sequence of pairwise different numbers $b_n \in \{0,1\}^{m(n)}$, $n \in \mathbb{N}$, such that

$$P \vdash_* \tau(C)_{b_n} \ .$$

The intuition is that for functions having pseudorandom properties it should be hard to prove that a given element lies outside the range of the function. The hardness of a p-stretching function can be characterized by a hitting set property for NP/poly-sets. For this we need the following definition of the resultant of a p-stretching map.

Definition 10 (Krajíček [4])**.** *Let f be a p-stretching map computed by the circuit family $C = (C_n)_{n\in\mathbb{N}}$ and let P be a propositional proof system. The* resultant of C with respect to P*, denoted by Res^P_C, consists of all* NP/poly*-sets A for which there exists a propositional representation $\varphi_n(\bar{x}, \bar{y})$ of A such that*

$$P \ \vdash_* \ \varphi_n(\bar{x}, \bar{y}) \to C(z) \neq x \ .$$

In [4] this definition is formulated slightly differently, but as already here the close connection to disjoint NP-pairs becomes visible we have used similar terminology as in the previous sections. The following theorem characterizes the hardness of τ-formulas by a condition on the resultant of P.

Theorem 11 (Krajíček [4])**.** *Let P be a proof system of the form $EF + \Phi$ for some polynomial-time computable set $\Phi \subseteq \mathrm{TAUT}$. Let f be a p-stretching function and*

C a polynomial-size circuit family computing f. Then the following conditions are equivalent:

1) *The formulas $\tau(C)$ are hard for P.*
2) *The resultant Res_C^P contains only finite sets.*

In fact, the hardness of the function f should not depend on the particular circuits used for the computation of f. For functions f computed by non-uniform circuit families it is, however, not possible to get hard formulas $\tau(C)$ for all circuit families C computing f.

While this is not difficult to prove formally it is also intuitively clear. If a function f is computed by the circuits C which might yield hard formulas $\tau(C)$, then we can modify these circuits to a circuit family C' as follows. To the output gates of C we attach a circuit of polynomial size which compares the output produced by C with polynomially many fixed elements from the complement of $\mathrm{rng}(f)$. If this test is positive, then we output a fixed element from $\mathrm{rng}(f)$, otherwise we return the original output of C. Obviously, C and C' compute the same function f. But intuitively the formulas $\tau(C')$ are not hard for sufficiently strong proof systems P. By inspecting the extra gates attached to the circuits C we can devise short P-proofs for the disjointness of $\mathrm{rng}(f)$ and the set of those elements which are excluded in the extra gates of C'.

However, the situation is different for the functions $f \in \mathsf{FP}$ which are computed by uniform circuit families. Focusing therefore on the case where the circuit families are uniformly given we say that a polynomial-time computable p-stretching function f yields *representationally independent hard τ-formulas* for P, if for every uniformly given circuit family C computing f the resulting formulas $\tau(C)$ are hard for P.

In this case also the resultant Res_C^P has to be defined efficiently and contains just NP-sets which are disjoint with $\mathrm{rng}(f)$ and where this disjointness is provable with short P-proofs. We can therefore use our terminology about disjoint NP-pairs to rephrase condition 2 of the theorem by the following condition 2':

2'. *All sets $A \in \mathsf{NP}$ with $(A, \mathrm{rng}(C)) \in \mathsf{DNPP}(P)$ are finite.*

We point out that in condition 2' the disjointness of A and $\mathrm{rng}(f)$ has to be proven with respect to the circuit family used for the computation of f, while the representation of A can be chosen arbitrarily.

Using the $\leq_s$-completeness of the W-pair for $\mathsf{DNPP}(P)$ (Theorem 9) we can restate Theorem 11 in the following form:

Corollary 12. *Let P be a proof system of the form $EF + \Phi$ for some polynomial-time computable set $\Phi \subseteq$ TAUT. For every p-stretching function $f \in \mathsf{FP}$ the following are equivalent:*

1) *f yields representationally independent hard τ-formulas for P.*
2) *Every set $A \in \mathsf{NP}$ with $A \cap \mathrm{rng}(f) = \emptyset$ and $(A, \mathrm{rng}(f)) \leq_s (W_1(P), W_2(P))$ is finite.*

The difference between Corollary 12 and Theorem 11 is that condition 2 of the corollary only speaks about $\mathrm{rng}(f)$ whereas condition 2 of the above theorem involves the particular circuits used for the computation of f.

Dropping the requirement $(A, \mathrm{rng}(f)) \leq_s W(P)$ from the second condition of Corollary 12 we arrive at an NP-set $B = \mathrm{rng}(f)$ containing no infinite NP-set in its complement $\bar{B}$. Such sets B are called NP-*simple* (see [24] or [36]). By Corollary 12, NP-simple sets would yield representationally independent hard τ-formulas for all proof systems, but their existence is open.

Simplicity is a concept originating in recursion theory that can be defined for any complexity class.

Definition 13. *Let C be a complexity class.*

1) *A set A is called* C-immune *if every subset $B \subseteq A$ with $B \in \mathsf{C}$ is finite.*
2) *A is called* C-simple*, if $A \in \mathsf{C}$ and $\bar{A}$ is* C*-immune.*

Here we are interested in the cases $\mathsf{C} = \mathsf{P}$ and $\mathsf{C} = \mathsf{NP}$. As mentioned, the question whether NP-simple sets exist is open. Obviously $\mathsf{NP} \neq \mathsf{coNP}$ is a necessary condition for the existence of NP-simple sets, other necessary or sufficient conditions are, however, not known. Vereshchagin proved that NP-simple sets exist relative to a random oracle [37].

What we actually need for the hardness of τ-formulas is not the existence of NP-simple sets, but a weaker condition which could be formalized as:

Definition 14. *Let (C, D) be a disjoint* NP*-pair. We call a set A* NP-simple relative to (C, D) *if $A \in \mathsf{NP}$ and for all infinite sets $B \in \mathsf{NP}$ with $A \cap B = \emptyset$ we have $(A, B) \not\leq_s (C, D)$.*

With this definition Corollary 12 takes the following form:

Corollary 15. *For all proof systems $P = EF + \Phi$ with polynomial-time computable $\Phi \subseteq$ TAUT and all p-stretching functions $f \in \mathsf{FP}$ the following are equivalent:*

1) *f yields representationally independent hard τ-formulas for P.*
2) *$\mathrm{rng}(f)$ is* NP*-simple relative to $(W_1(P), W_2(P))$.*

The following easy proposition gives a characterization of the relative simplicity of an NP-set.

Proposition 16. *Let $A \in \mathsf{NP}$ and let (C, D) be a disjoint* NP*-pair. Then A is* NP*-simple relative to (C, D) if and only if for all $\leq_m^p$-reductions $g : A \leq_m^p C$ the set $g^{-1}(D)$ is finite.*

Proof: Let A be NP-simple relative to (C, D). Let us assume that $g^{-1}(D)$ is infinite for some reduction $g : A \leq_m^p C$. We have $g^{-1}(D) \in \mathsf{NP}$ and $A \cap g^{-1}(D) = \emptyset$. Therefore g reduces the disjoint NP-pair $(A, g^{-1}(D))$ to (C, D), i.e. A is not NP-simple relative to (C, D).

If on the contrary A is not NP-simple relative to (C, D), then there exists an infinite set $B \in \mathsf{NP}$ with $A \cap B = \emptyset$ and $g : (A, B) \leq_s (C, D)$ via some function $g \in \mathsf{FP}$. Then $g^{-1}(D)$ contains B and is therefore infinite. ■

The proof of Proposition 16 also makes it clear that the relative NP-simplicity of a set does not depend on the strength of the reduction used, i.e. using the weaker reduction $\leq_p$ instead of $\leq_s$ in Definition 14 results in the same concept.

In view of the above proposition the NP-simplicity of A relative to (C, D) can also come from the fact that A is not $\leq_m^p$-reducible to C. But for the case where $(C, D) = (W_1(P), W_2(P))$ this cannot happen as $W_1(P)$ and $W_2(P)$ are NP-complete. In this case we can give the following necessary condition for the relative NP-simplicity of A.

Proposition 17. *Let A be NP-simple relative to (C, D) and let A be $\leq_m^p$-reducible to C. Then $\bar{A}$ is P-immune.*

Proof: Let $g : A \leq_m^p C$. If $\bar{A}$ is not P-immune, then there exists an infinite set $B \in \mathsf{P}$ with $A \cap B = \emptyset$. Then the disjoint NP-pair (A, B) is $\leq_s$-reducible to (C, D) via

$$g'(x) = \begin{cases} g(x) & \text{if } x \notin B \\ x_0 \in D & \text{if } x \in B, \end{cases}$$

i.e. A is not NP-simple relative to (C, D). ■

Therefore the relative NP-simplicity of a set A is a notion which lies in strength between the P-immunity of the complement $\bar{A}$ and the NP-simplicity of A. Whether disjoint NP-pairs will indeed prove to be helpful in establishing lower bounds to the proof size in strong proof systems must remain open. The characterization of these difficult proof-theoretic problems in terms of disjoint NP-pair as given in Corollary 12 shows, however, that investigation into the structure of NP-pairs will remain a demanding and potentially rewarding task.

Acknowledgments

I thank Jan Krajíček and Zenon Sadowski for helpful discussions on the topic of this paper.

References

[1] J. Grollmann and A. L. Selman, "Complexity measures for public-key cryptosystems," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 309–335, 1988.

[2] A. A. Razborov, "On provably disjoint NP-pairs," Electronic Colloquium on Computational Complexity, Technical Report TR94-006, 1994.

[3] P. Pudlák, "On reducibility and symmetry of disjoint NP-pairs," *Theoretical Computer Science*, vol. 295, no. 1–3, pp. 323–339, 2003.

[4] J. Krajíček, "Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds," *The Journal of Symbolic Logic*, vol. 69, no. 1, pp. 265–286, 2004.

[5] J. Krajíček and P. Pudlák, "Some consequences of cryptographical conjectures for S_2^1 and EF," *Information and Computation*, vol. 140, no. 1, pp. 82–94, 1998.

[6] C. Glaßer, A. L. Selman, and L. Zhang, "Canonical disjoint NP-pairs of propositional proof systems," *Theoretical Computer Science*, vol. 370, no. 1–3, pp. 60–73, 2007.

[7] ——, "The informational content of canonical disjoint NP-pairs," *International Journal of Foundations of Computer Science*, to appear.

[8] O. Beyersdorff, "Tuples of disjoint NP-sets," *Theory of Computing Systems*, vol. 43, no. 2, pp. 118–135, 2008.

[9] ——, "Classes of representable disjoint NP-pairs," *Theoretical Computer Science*, vol. 377, no. 1–3, pp. 93–109, 2007.

[10] ——, "The deduction theorem for strong propositional proof systems," *Theory of Computing Systems*, to appear.

[11] Z. Sadowski, "Optimal proof systems, optimal acceptors and recursive presentability," *Fundamenta Informaticae*, vol. 79, no. 1–2, pp. 169–185, 2007.

[12] C. Glaßer, A. L. Selman, and L. Zhang, "Survey of disjoint NP-pairs and relations to propositional proof systems," in *Essays in Theoretical Computer Science in Memory of Shimon Even*, O. Goldreich, A. L. Rosenberg, and A. L. Selman, Eds. Springer-Verlag, Berlin Heidelberg, 2006, pp. 241–253.

[13] C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang, "Disjoint NP-pairs," *SIAM Journal on Computing*, vol. 33, no. 6, pp. 1369–1416, 2004.

[14] C. Glaßer, A. L. Selman, and S. Sengupta, "Reductions between disjoint NP-pairs," *Information and Computation*, vol. 200, no. 2, pp. 247–267, 2005.

[15] J. H. Lutz and E. Mayordomo, "Inseparability and strong hypotheses for disjoint NP pairs," Electronic Colloquium on Computational Complexity, Technical Report TR09-022, 2009.

[16] J. Köbler, J. Messner, and J. Torán, "Optimal proof systems imply complete sets for promise classes," *Information and Computation*, vol. 184, no. 1, pp. 71–92, 2003.

[17] S. A. Cook and J. Krajíček, "Consequences of the provability of NP $\subseteq$ P/poly," *The Journal of Symbolic Logic*, vol. 72, no. 4, pp. 1353–1371, 2007.

[18] O. Beyersdorff and S. Müller, "Does advice help to prove propositional tautologies?" in *Proc. 12th International Conference on Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, vol. 5584. Springer-Verlag, Berlin Heidelberg, 2009, pp. 65 – 72.

[19] J. Krajíček, "On the weak pigeonhole principle," *Fundamenta Mathematicae*, vol. 170, pp. 123–140, 2001.

[20] ——, "Tautologies from pseudo-random generators," *Bulletin of Symbolic Logic*, vol. 7, no. 2, pp. 197–212, 2001.

[21] ——, "A proof complexity generator," in *Proc. 13th International Congress of Logic, Methodology and Philosophy of Science*, ser. Studies in Logic and the Foundations of Mathematics. King's College Publications, London, 2007.

[22] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, "Pseudorandom generators in propositional proof complexity," *SIAM Journal on Computing*, vol. 34, no. 1, pp. 67–88, 2004.

[23] A. A. Razborov, "Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution," 2003, preprint.

[24] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I*. Springer-Verlag, Berlin Heidelberg, 1988.

[25] S. A. Cook and R. A. Reckhow, "The relative efficiency of propositional proof systems," *The Journal of Symbolic Logic*, vol. 44, no. 1, pp. 36–50, 1979.

[26] J. Krajíček and P. Pudlák, "Propositional proof systems, the consistency of first order theories and the complexity of computations," *The Journal of Symbolic Logic*, vol. 54, no. 3, pp. 1063–1079, 1989.

[27] A. L. Selman, "A taxonomy of complexity classes of functions," *Journal of Computer and System Sciences*, vol. 48, no. 2, pp. 357–381, 1994.

[28] 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*, ser. Lecture Notes in Computer Science, vol. 1974. Springer-Verlag, Berlin Heidelberg, 2000, pp. 361–372.

[29] U. Schöning, "A low and a high hierarchy within NP," *Journal of Computer and System Sciences*, vol. 27, pp. 14–28, 1983.

[30] K. Ko and U. Schöning, "On circuit-size complexity and the low hierarchy in NP," *SIAM Journal on Computing*, vol. 14, pp. 41–51, 1985.

[31] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[32] H. G. Rice, "Classes of recursively enumerable sets and their decision problems," *Trans. Am. Math. Soc.*, vol. 74, pp. 358–366, 1953.

[33] J. Krajíček, *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, ser. Encyclopedia of Mathematics and Its Applications. Cambridge: Cambridge University Press, 1995, vol. 60.

[34] O. Beyersdorff, "On the correspondence between arithmetic theories and propositional proof systems – a survey," *Mathematical Logic Quarterly*, vol. 55, no. 2, pp. 116–137, 2009.

[35] A. Atserias and M. L. Bonet, "On the automatizability of resolution and related propositional proof systems," *Information and Computation*, vol. 189, no. 2, pp. 182–201, 2004.

[36] T. Yamakami and T. Suzuki, "Resource bounded immunity and simplicity," *Theoretical Computer Science*, vol. 347, no. 1–2, pp. 90–129, 2005.

[37] N. K. Vereshchagin, "NP-sets are Co-NP-immune relative to a random oracle," in *Proc. 3rd Israel Symposium on Theory of Computing and Systems*, 1995, pp. 40–45.