Survey of Disjoint NP-Pairs and Propositional Proof Systems

by

Nils Wisiol

September, 1st 2014

A thesis submitted to the Faculty of the Graduate School of the University at Buffalo, State University of New York in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science and Engineering

I wish to thank my advisors Alan Selman (University at Buffalo) and Christian Glaßer (Universität Würzburg, Germany), committee member Ken Regan and my friends and fellow students Andrew Hughes and Michael Wehar for their great support, patience, knowledge and comments.

Contents

This thesis on propositional proof systems and disjoint NP-pairs gives a survey of these fields. We present history and motivation of both theories by giving examples for their use. The reader is then introduced into the formal notions of the fields. Dedicated chapters present important and outstanding results from the theories. Some results are proven, some results are given without a proof. It follows a chapter that presents the relation of both fields with a result due to Razborov. As for none of the assertions in this thesis the absolute truth value is known, we also survey some oracles relative to which we know the truth value of important statements. We finally look into open questions and suggest future work on both fields.

1 Introduction

This thesis aims at readers that do not have a strong background in the theories of propositional proof systems and disjoint NP pairs. It surveys important results from both and points out important connections in between these two theories. Core of the thesis is the implication chart in Figure 1.1, that summarizes virtually all results mentioned in this thesis.

In Section 2 is split into two pieces. In 2.1, we introduce the reader to the theory of disjoint NP-pairs based on the notion of promise problems. In 2.2, we familiarize ourselves with propositional proof systems. Both introductions contain history, motivation and notions of the theories. In Section 2.2, we give basic results and proofs that help the reader understanding the introduced notions. Readers familiar with notions from both fields can safely skip this Section.

Section 3 covers important results from the field of disjoint NP-pairs. We will study the earlier introduced reducibility of pairs in greater detail. It turns out that various definitions of reducibility available in the literature are equivalent. Subsequently, we study refinements of the ESY-conjecture and connections to open questions of complexity class separation.

Greater details of the theory of propositional proof systems will be covered in Section 4. We will justify the motivation to study proof systems by establishing an equivalent formulation of NP = coNP, before we look into sufficient and necessary conditions for their existence. These conditions will lay the foundation for the study of different oracles in Section 6.

Section 5 finally covers the connection in between both theories that was discovered by Razborov. It presents a proof for Razborov's theorem that uses the notion of complexity theory.

We also take a look on relativized worlds in Section 6. We point the reader to oracles relative to which optimal proof systems exist respectively do not exist. Section 6.2 studies the converse of Razborov's theorem, for which we know oracles relative to which it holds, and relative to which it does not hold. Finally we refer the reader to a oracle that separates different refinements of the ESY-conjecture from each other.

We conclude the thesis with a summary of open questions and future work on the field in Section 7.

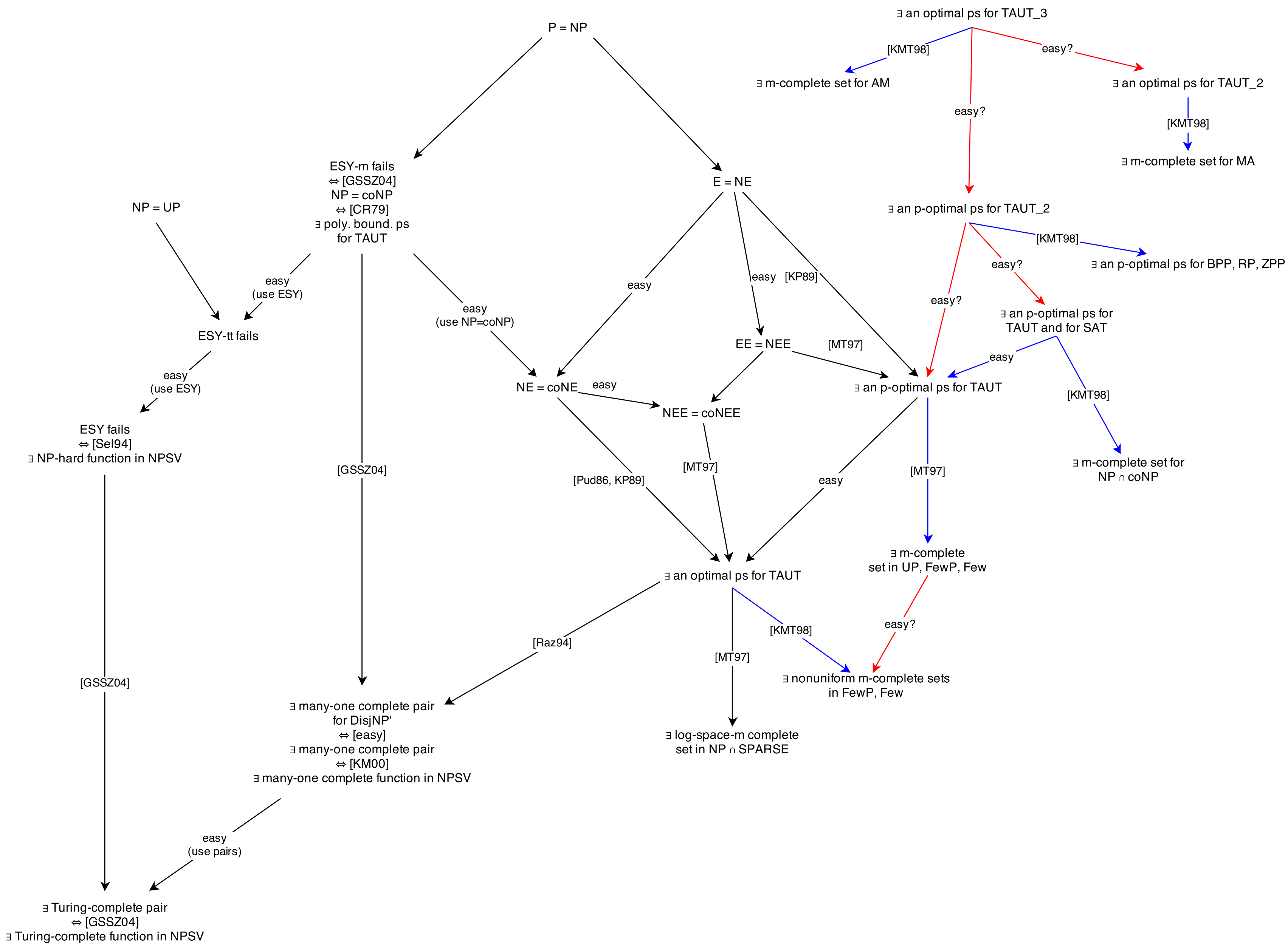

Figure 1.1: Known Implications for proof systems, disjoint pairs and the ESY conjecture

2 Preliminaries

2.1 Disjoint NP-Pairs

The study of disjoint NP-pairs originates in the study of public key crypto systems (PKCS). The interest in secure PKCS is fundamental to everyday life as well as to academia, as provably hard-to-crack PKCS would imply NP $\neq$ P.

To study the hardness of PKCS, Even, Selman and Yacobi used the notion of *promise problems* rather than decision problems to model the problem of cracking a PKCS [ESY84]. In fact, promise problems are a generalization of decision problems. A machine working on a promise problem is not only given an input, but also a promise that for this input, a certain condition holds. The machine *solves* the problem, if it gives the right answer on all inputs for which the promise holds. If the promised condition does in fact not hold for a given input, then the machine can act arbitrarily.

We can define promise problems more formally, following Goldreich's survey [Gol05]: A promise problem is a partition of the set of all strings into three subsets:

1. The set of strings representing Yes-Instances,
2. the set of strings representing No-Instances, and
3. the set of disallowed strings.

We can write this partition as a pair of two disjoint sets (A, B), where A and B represent Yes- and No-Instances, and the set of disallowed strings is $\overline{A \cup B}$. The *promise* in this setting is that a given input string either belongs to A or B. If $\overline{A \cup B} = \emptyset$, then the promise problem has no disallowed strings and thus no promise, it is in fact a decision problem.[1]

[1] Even, Selman and Yacobi [ESY84] used a pair (Q, R) to represent promise problems, where Q is a predicate true for all *allowed* strings (the *promise*) and R is a predicate true for all Yes-Instances (the *property*). This relates with Goldreich's definition as follows:

$$\begin{aligned} A \cup B &= \{w \in \Sigma^* \mid Q(w)\} \\ A &= \{w \in \Sigma^* \mid Q(w) \wedge R(w)\} \\ B &= \{w \in \Sigma^* \mid Q(w) \wedge \neg R(w)\} \end{aligned}$$

Using this notation, we can define a promise problem that captures the hardness of cracking the PKCS, that is, captures the hardness of finding the plain text to a given cipher text C and public key K. To crack the crypto system, we will conduct a binary search among all strings up to a reasonable length. The scope for the binary search is limited, as the length of the plain text is polynomial in the length of the cipher text. Notice that this notion captures the hardness to crack *every* cipher text in a PKCS. While we can conclude cryptographic insecureness from an easy cracking problem, a hard cracking problem does not imply cryptographic security, as a subset of cipher texts may be still easy to crack.

The promise problem is defined as follows:

1. The set of Yes-Instances will be the set of strings $\langle M', C, K\rangle$ for which there exists a message M, $M \leq M'$, such that M encrypted with K yields cipher text C.
2. The set of No-Instances will be the set of strings $\langle M', C, K\rangle$ for which there exists a message M, $M > M'$, such that M encrypted with K yields cipher text C.
3. The set of disallowed strings will be all triples $\langle M', C, K\rangle$ such that for all plain texts M, encryption with K does not yield C.

With a machine solving this promise problem, we can find the plain text to any given C and K by binary search over all messages M'. Thus, the runtime of cracking the PKCS is within a logarithmic factor of the runtime of the machine solving the promise problem. Therefore, we consider the hardness of the promise problem itself as a good measurement for the hardness of the cracking problem.

But why not model the cracking problem as a decision problem? To see why a simple decision problem does not capture the cracking problem correctly, assume we have a PKCS such that the *decision* problem A is not efficiently computable. However, if there is an algorithm efficiently solving the promise problem (A, B), the crypto system would still be easy to crack. On the other hand, if the promise problem is hard, the decision problem will also be.

To capture the situation where we have a hard decision problem, but an easy promise problem, we call a set S for which $A \subseteq S$ and $B \subseteq \overline{S}$ *a separator.* Let the set $\mathrm{Sep}(A, B)$ denote the set of all separators for a given pair (A, B). A pair (A, B) that has no polynomial-time decidable set in $\mathrm{Sep}(A, B)$ is called P-*inseparable*, otherwise it is called P-*separable*.

The interesting class of promise problems (A, B) is the class with promises that are not polynomial-time decidable. In the contrary case where $A \cup B$ is efficiently computable, deciding the promise problem (A, B) is polynomial-time equivalent to solving the decision problems A or B.

Assigning a hardness to PKCS immediately calls for a notion that compares the hardness of two promise problems. Following Grollmann and Selman [GS88], we use the following reductions for promise problems that naturally arise from the reductions of languages.

Definition 1.

1. A promise problem (A, B) is *many-one-reducible in polynomial time* to (C, D), $(A, B) \leq_m^{pp} (C, D)$, if for every separator $T \in \mathrm{Sep}(C, D)$, there exists a separator $S \in \mathrm{Sep}(A, B)$ such that $S \leq_m^p T$.

2. A promise problem (A, B) is *many-one-reducible in polynomial time* to (C, D), $(A, B) \leq_T^{pp} (C, D)$, if for every separator $T \in \mathrm{Sep}(C, D)$, there exists a separator $S \in \mathrm{Sep}(A, B)$ such that $S \leq_T^p T$.

3. As a generalization of the previous two, we define a promise problem (A, B) to be *r-reducible* to (C, D), $(A, B) \leq_r^{pp} (C, D)$, if for every separator $T \in \mathrm{Sep}(C, D)$, there exists a separator $S \in \mathrm{Sep}(A, B)$ such that $S \leq_r T$.

4. A promise problem (A, B) is NP-hard, if for every Turing machine M that solves (A, B), the language accepted by M is NP-hard.

A promise problem (A, B) with $A, B \in \mathrm{NP}$ is a *disjoint* NP-*pair*. We define DisjNP to be the set of all disjoint NP-pairs. For this class, we define completeness:

Definition 2.

1. A disjoint NP-pair (A, B) is $\leq_m^{pp}$-complete, if for every $(C, D) \in \mathrm{DisjNP}$ we have $(C, D) \leq_m^{pp} (A, B)$.

2. A disjoint NP-pair (A, B) is $\leq_T^{pp}$-complete, if for every $(C, D) \in \mathrm{DisjNP}$ we have $(C, D) \leq_T^p (A, B)$.

We define $\leq_r^{pp}$-completeness analogously.

Evan, Selman and Yacobi found out that if disjoint NP-pairs that are NP-hard do not exist, then there is exist no PKCS with NP-hard cracking problems. This gives additional motivation for the study of disjoint NP-pairs, which also arise from the study of recursively inseparable sets.

The assertion that there are no disjoint NP-pairs that are NP-hard to solve has many more consequences and has been studied well since it was formulated as a conjecture by Even, Selman, and Yacobi [ESY84].

Conjecture 3 (ESY)**.** *For every pair of disjoint sets in* NP*, there is a separator that is not Turing-hard for* NP*. [ESY84]*

If the conjecture holds, then no PKCS is NP-hard to crack. The following refined version of the ESY-conjecture can be proven to be equivalent to NP $\neq$ coNP, see Theorem 15.

Conjecture 4 (ESY-m)**.** *For every pair of disjoint sets in* NP*, there is a separator that is not many-one-hard for* NP*. [HPRS12]*

We will study the consequences of the ESY-conjectures in Section 3.2.

2.2 Propositional Proof Systems

A propositional proof system is a fast method of verifying proofs.

To give an example, we will have a look at the *resolution principle*, which was introduced by Robinson [Rob65]. Consider a boolean formula φ in conjunctive normal form. If φ is not satisfiable, the resolution principle provides a way to find a proof for this fact. To find a proof, the resolution principle iteratively combines two existing clauses into a new and shorter clause with equivalent truth value. Robinson showed that the resolution principle yields the empty clause eventually for any unsatisfiable formula, and any formula for which the principle yields the empty clause is unsatisfiable:

Theorem 5 (Resolution Theorem [Rob65])**.** *For a formula φ in conjunctive normal form, the resolution principle yields the empty clause if and only if φ is not satisfiable.*

As we can see from the way resolution works, there are exponentially many options how to combine the clauses, and not every sequence of combinations will yield the empty clause. Hence, it is hard to find a sequence of combinations that derive the empty clause. By Theorem 5, this sequence exists if and only if the formula is unsatisfiable. As opposed to finding a sequence, given a sequence of combinations, we can easily check if this sequence derives the empty clause.

Using formal terms, let f be defined by

$$f(\langle\varphi, w\rangle) = \begin{cases} \neg\varphi & \text{if combination sequence } w \text{ applied to } \varphi \text{ yields the empty clause,} \\ \bot & \text{otherwise.} \end{cases}$$

Intuitively, f is polynomial-time computable. By the Resolution Theorem, f only outputs tautologies, and for every tautology $\neg\varphi$, there is an input $\langle\varphi, w\rangle$ such that $f(\langle\varphi, w\rangle) = \neg\varphi$.

Given a combination sequence w that yields the empty clause for φ, the function f provides a fast way to verify $\neg\varphi$ is a tautology. We thus call f a propositional proof system, and we call $\langle\varphi, w\rangle$ a f-proof for $\neg\varphi$.

Definition 6. A polynomial-time computable function f that is onto the set of tautologies is called a *propositional proof system* or *proof system*. For any w, we say w is a *f-proof for x* if $f(w) = x$. If there is a polynomial p, such that for all x, and all f-proofs w of x, we have $|w| \leq p(|x|)$, then f is *polynomially-bounded*.

Cook and Reckhow started a line of research [CR79] that tries to investigate what the length of the shortest proof of a propositional tautology relative to the length of the tautology is. The interest in the length of the proof is motivated by the fact that the existence of polynomial-length proofs for all tautologies characterizes the question of whether NP = coNP. (A fact we will prove in Section 4.) However, no known proof system could be proven to have proofs with length bounded by a polynomial. To tackle the problem, Cook and Reckhow introduced the notion of simulation of proof systems.

Definition 7. Let f and g be proof systems. We say f *simulates* g, if there is a function h such that for all w, it holds $f(h(w)) = g(w)$ and $|h(w)| \leq p(|w|)$. If h is polynomial-time computable, we say f *p-simulates* g. A proof system that simulates (p-simulates) every other proof system is called *optimal* (*p-optimal*).

An more intuitive (and informal) way to give a definition for "f simulates g" is to say that for every tautology φ, the f-proof for φ is at most polynomially longer than the g-proof of φ. An optimal proof system then has the shortest proofs for tautologies among all proof systems, within a polynomial factor.

However, it is not only unknown whether polynomially-bounded proof systems exist, it is also unknown if optimal or even p-optimal proof systems exist. To study the existence of optimal and p-optimal proof systems, we will therefore study sufficient conditions and implications in Section 4. To become familiar with the notions, we present a strong sufficient condition for the existence of optimal proof systems:

Theorem 8. *If* $\mathrm{NP} = \mathrm{coNP}$*, then there is an optimal proof system.*

Proof. Let N be a NP-machine deciding $\mathrm{TAUT} \in \mathrm{coNP}$. We define f by

$$f(\langle i, x \rangle) = \begin{cases} x & \text{if } N \text{ accepts } w \text{ along path } i, \\ \perp & \text{otherwise.} \end{cases}$$

Notice, a proof system does not have to be a total function. By definition, f outputs only tautologies, and for every tautology there is an accepting path of N, so f is onto TAUT.

To see f is optimal, let f' be an arbitrary proof system. There is a function g mapping each tautology w to an accepting path i of N. Notice, g might not by polynomial-time computable, but is polynomially length bounded. Let now w be a f'-proof for x. With g, we can translate w into $\langle g(w), f'(w) \rangle$, which is a f-proof for x. □

As we will see in Section 4, the existence of both optimal and p-optimal proof systems can be proven under much weaker hypotheses.

3 Disjoint NP-Pairs

One of the most interesting open questions about disjoint NP-pairs is whether there are complete pairs, either $\leq_m^{pp}$- or $\leq_T^{pp}$-complete. A proof of the non-existence of either one would prove NP $\neq$ coNP and P $\neq$ NP, but we can also relate complete pairs to propositional proof systems. Using the ESY-conjectures, we can also relate disjoint NP-pairs to the NP vs. coNP questions.

3.1 Reducibility of disjoint pairs

In the literature exist several different definitions for the reducibility of pairs. Notice that results from this section apply to all disjoint pairs (A, B); the sets are *not* required to be in NP. Additionally to the definition 1 given above in the introduction, Grollmann and Selman [GS88] also define the notion of *uniform* reductions of pairs:

Definition 9. Let (A, B) and (C, D) be disjoint pairs.

1. (A, B) is *uniformly many-one reducible in polynomial time* to (C, D), $(A, B) \leq_{um}^{pp} (C, D)$, if there exists a polynomial-time computable function f such that for every separator $T \in \mathrm{Sep}(C, D)$, there exists a separator $S \in \mathrm{Sep}(A, B)$ such that $S \leq_m^p T$ via f.

2. (A, B) is *uniformly Turing reducible in polynomial time* to (C, D), $(A, B) \leq_{uT}^{pp} (C, D)$, if there exists a polynomial-time oracle Turing machine M such that for every separator $T \in \mathrm{Sep}(C, D)$, there exists a separator $S \in \mathrm{Sep}(A, B)$ such that $S \leq_T^p T$ via M.

Notice that this definition requires that all separators reduce via the same function or machine. Definition 1, the definition of nonuniform reducibility, does not require this. In spite of this, surprisingly, it turns out that the uniform and nonuniform variant of the definition are equivalent.

Razborov uses yet another definition of many-one reducibility of pairs:

Definition 10. Let (A, B) and (C, D) be disjoint pairs. (A, B) is Razborov reducible[2] to (C, D), if there exists a polynomial-time computable function λ such that $\lambda(A) \subseteq C$ and $\lambda(B) \subseteq D$.

[2]Razborov reducible is not a term commonly used in the literature. We will use it only to prove equivalence to many-one reducibility in Lemma 11.

It turns out that this is equivalent to the many-one reducibility defined above as well. As a summary of all these definitions, we obtain the following lemma. A comprehensive proof of it can be found Glaßer, Selman, Sengupta and Zhang [GSSZ03, Theorems 2.8, 2.10, 2.14].

Lemma 11. *Let (A, B) and (C, D) be disjoint pairs. Then the following assertions are equivalent:*

1. $(A, B) \leq_m^{pp} (C, D)$
2. $(A, B) \leq_{um}^{pp} (C, D)$
3. *There exists a polynomial-time computable function λ such that $\lambda(A) \subseteq C$ and $\lambda(B) \subseteq D$.*

The assertions above imply the following equivalent statements:

1. $(A, B) \leq_T^{pp} (C, D)$
2. $(A, B) \leq_{uT}^{pp} (C, D)$

Therefore, for the rest of this thesis, we will only use many-one and Turing reducibility.

3.2 ESY-conjectures

The original ESY-conjecture [ESY84] is that for every pair of disjoint sets in NP, there is a separator that is not Turing-hard for NP. This can easily be refined by using many-one hardness instead of Turing-hardness.

Definition 12. For a reduction r, we define the ESY-r conjecture as follows: For every pair of disjoints sets in NP, there is a separator that is not r-hard for NP,

$$\forall_{(A,B)\in \mathrm{DisjNP}} \exists_{S \in \mathrm{Sep}(A,B)} \exists_{L \in \mathrm{NP}} L \not\leq_r S.$$

Notice, ESY-T is the original ESY conjecture.

The negation of the ESY-r conjecture is

$$\exists_{(A,B)\in \mathrm{DisjNP}} \forall_{S \in \mathrm{Sep}(A,B)} \forall_{L \in \mathrm{NP}} L \leq_r S,$$

that is, there exists a disjoint NP-pair (A, B) such that all separators are r-hard for NP. Since the different reductions imply each other, we obtain a implication chain of ESY-conjectures:

Lemma 13. *Each item implies the following item in the list:*

1. *ESY-m does not hold.*
2. *ESY-tt does not hold.*
3. *ESY-T (the original ESY conjecture) does not hold.*

This list can, of course, be extended to a lot more reductions of languages. In this thesis, we mention these specific reductions because there are known results that relate to these assertions.

The ESY-conjectures immediately relate to the existence of complete pairs, as we can see from the negated ESY-r statement.

Theorem 14. *If ESY-r does not hold, then there exists a r-complete disjoint* NP *pair.*

Proof. Assume that ESY-r does not hold, then there is a pair (A, B) of disjoint sets in NP such that every separator is r-hard for NP. We claim (A, B) is r-complete for DisjNP. Let $(C, D) \in$ DisjNP, and let S be any separator for (A, B). Then $C \in$ NP and S is r-hard for NP, $C \leq_r S$. This proves C, which is a separator of (C, D), reduces to any separator of (A, B). By definition 1, we have $(C, D) \leq_r^{pp} (A, B)$ and thus (A, B) is r-complete for DisjNP. ☐

As mentioned above, the refinements of the (original) ESY-T have interesting relations to computational complexity as well. ESY-m connects to the NP vs. coNP question, and ESY-tt relates to the question whether NP = UP.

Theorem 15. *[GSSZ03]The following assertions are equivalent.*

1. *The ESY-m conjecture does not hold, that is, there exists a disjoint* NP*-pair such that all separators are many-one-hard for* NP*.*
2. NP = coNP

Proof. ☐

Theorem 16. *If* NP = UP*, then ESY-tt does not hold, that is there exists a disjoint* NP*-pair such that all separators are truth-table-hard for* NP*.*

Proof. □

3.3 more

Razborov [Raz94] discovered that the existence of optimal proof systems implies the existence of complete pairs, as we will see in Corollary 22.

Are there P-inseparable pairs? This implies P $\neq$ NP.

4 Propositional Proof Systems

4.1 Polynomially-bounded proof systems and NP = coNP

We will show that the existence of polynomially-bounded proof systems characterizes the statement NP = coNP. The proof is due to Cook and Reckhow [CR79].

Theorem 17. *There is a polynomially-bounded propositional proof system if and only if* NP = coNP.

Proof. Assume NP = coNP and let M be an NP-machine accepting TAUT. We define a proof system f, in which all proofs are polynomially length bounded:

$$f(\langle\varphi, w\rangle) = \begin{cases} \varphi & \text{if } w \text{ is an accepting path of } M \text{ on input } \varphi, \\ \text{true} & \text{otherwise.} \end{cases}$$

Since f only considers one path in the computation of M, it is polynomial-time computable. Also, f only outputs tautologies. Therefore, f is a proof system. As there is an accepting path in the computation of M for every tautology φ, all tautologies have polynomial-length proofs.

To prove the converse, suppose there is a polynomially-bounded proof system f. Since the complement of TAUT is NP-complete, it suffices to show TAUT $\in$ NP. Let M be a nondeterministic Turing machine such that on input φ, M guesses an f-proof w of polynomial length and calculates $f(w)$. The machine then accepts if and only if $f(w) = \varphi$. Hence, M is an NP-machine accepting TAUT. □

Cook and Reckhow introduced the notion of optimal proof systems in order to prove NP $\neq$ coNP, that is, to prove there is no polynomially-bounded proof system. We call a proof system f optimal, if f-proofs are the shortest proofs among all proof systems (with respect to a polynomial factor). Proving the existence of an optimal proof system with proofs that are not within polynomial length shows NP $\neq$ coNP.

The existence of both polynomially bounded and optimal proof systems is unknown. However, we are able to prove some necessary and sufficient conditions.

4.2 Sufficient Conditions for the Existence of Optimal Proof Systems

To investigate further the question of whether optimal or even p-optimal proof systems exist, first Krajíček and Pudlák [KP89] and later Meßner and Torán proved sufficient conditions for the existence of such proof systems. The results reveal a symmetry for sufficient conditions for optimal and p-optimal proof systems:

$$
\begin{tikzcd}
\mathrm{P} = \mathrm{NP} \arrow[r] \arrow[d] & \mathrm{E} = \mathrm{NE} \arrow[r] \arrow[d] & \mathrm{EE} = \mathrm{NEE} \arrow[r] \arrow[d] & \begin{array}{c}\exists \text{ an p-optimal} \\ \text{proof system}\end{array} \arrow[d] \\
\mathrm{NP} = \mathrm{coNP} \arrow[r] & \mathrm{NE} = \mathrm{coNE} \arrow[r] & \mathrm{NEE} = \mathrm{coNEE} \arrow[r] & \begin{array}{c}\exists \text{ an optimal} \\ \text{proof system}\end{array}
\end{tikzcd}
$$

Figure 4.1: The symmetric structure of sufficient conditions for optimal and p-optimal propositional proof systems.

We call a language L *almost-tally*, if every string in L is of the form 0^*10^*. By $\mathcal{P}(0^*10^*)$ we denote the class of all almost-tally languages. Meßner and Torán use the notion of almost-tally languages to obtain an even weaker sufficient condition than mentioned in the chart:

Theorem 18. 1. *If all almost-tally languages in* NEE *also belong to* EE*, then there exists a p-optimal propositional proof system.*

2. *If all almost-tally languages in* coNEE *also belong to* NEE*, then there exists an optimal propositional proof system.*

For the proof of 18.1, please refer to the original paper by Meßner and Torán [MT97].

The proof of 18.2 is based on constructing the almost-tally language T that belongs to coNEE. By the hypothesis, we can then assume $T \in \mathrm{EE}$ and $T \in \mathrm{NEE}$ respectively and define a proof system based on T.

Proof of 18.2. Let $M_1, M_2, ...$ be an enumeration of deterministic Turing transducers such that there is a universal Turing transducer that can simulate k steps of M_i in $(ik)^2$ steps. Define the almost-tally language

$$T = \{0^j 10^i \quad | \quad \text{for all words } w \text{ of length at most } 2^{2^{j+1+i}}:$$
$$\text{(if } M_i \text{ halts on } w \text{ in at most } 2^{2^{j+1+i}} \text{ steps, then } M_i \text{ outputs a tautology)}\}.$$

To see that T is a coNEE-language, we rewrite T as

$$T = \{0^j 10^i \quad | \quad \forall w, y \in \Sigma^{\leq 2^{2^{j+1+i}}} :$$
$$\left[M_i(w) \text{ halts in } 2^{2^{j+1+i}} \text{ steps with output } \varphi \implies \varphi(y) = \text{true})\right]\},$$

where the condition written in square brackets can be decided in deterministic double-exponential time. By the hypothesis, we thus have $T \in \text{NEE}$. Let N_T denote the nondeterministic Turing machine deciding T in time 2^{c2^n}, $c \geq 1$.

Based on N_T, we will now define a proof system f,

$$f(\langle 0^j 10^i, 0^s, \alpha, w\rangle) = \begin{cases} M_i(w) & \text{if for } l = j+1+i \text{ all of the following requirements are met:} \\ & \text{(a) } s \geq 2^{2^l}, \\ & \text{(b) } |w| \leq 2^{2^l}, \\ & \text{(c) } M_i \text{ on input } w \text{ halts in at most } 2^{2^l} \text{ steps,} \\ & \text{(d) } \alpha \text{ is an accepting computation of } N_T \text{ on input } 0^j 10^i, \\ \text{true} & \text{otherwise.} \end{cases}$$

First, we will show that f is a proof system. In both cases of the definition, f only outputs tautologies. Also, for any given tautology φ, there is a k such that M_k outputs φ on any input with length at least $|\varphi|$, and true for all shorter inputs. Hence, $10^k \in T$. Therefore, there is an α such that $\langle 10^k, 0^{2^{2^{k+1}}}, \alpha, 0^{|\varphi|}\rangle$ is a f-proof for φ. This confirms $f(\Sigma^*) = \text{TAUT}$. As a last condition, we need to verify f is polynomial-time computable: a machine computing f first checks if $s \geq 2^{2^l}$. If this is true, conditions (b), (c) and (d) can be verified in polynomial-time in $|0^s|$. If the check exceeds the polynomial-time limit, condition (a) does not hold and true will be output. Hence, f is polynomial-time computable.

To demonstrate that f is an optimal proof system, let g be any other proof system. For a given g-proof w, where g is computed by transducer M_i with time bound $n^k + k$, we will see that there is an α such that

$$\begin{aligned} w' &= \langle 0^j 10^i, 0^s, \alpha, w \rangle, \text{ where} \\ s &= 2^{c 2^{j+1+i}} \\ j &= \max(0, \left\lceil \log\log \left(|w|^k + k \right) \right\rceil - i - 1) \end{aligned}$$

is an f-proof for the same tautology, because the string w' satisfies all conditions in the first case of the definition of f, and therefore $f(w') = M_i(w) = g(w)$: (a) is satisfied by the choice of s, (b) holds in both of the following cases by choice of j.

$$\text{If } j > 0, \qquad 2^{2^l} \geq 2^{2^j} = 2^{2^{\left\lceil \log\log\left(|w|^k+k\right) \right\rceil}} \geq 2^{2^{\log\log\left(|w|^k+k\right)}} = |w|^k + k \geq |w|.$$

$$\text{If } j = 0, \qquad \left\lceil \log\log \left(|w|^k + k \right) \right\rceil - i - 1 \leq 0 \implies \left\lceil \log\log \left(|w|^k + k \right) \right\rceil \leq i + 1$$

$$\implies \log\log \left(|w|^k + k \right) \leq i + 1 \implies |w|^k + k \leq 2^{2^{i+1}} = 2^{2^l} \implies |w| \leq |w|^k + k \leq 2^{2^l}.$$

Condition (c), again, holds by choice of j: The runtime of M_i on input w is bounded by $|w|^k + k$, which is, as we have just seen, in both cases less or equal than 2^{2^l}. For condition (d), remember that M_i is computing a proof system and thus only outputs tautologies, which implies $0^j 10^i \in T$. Therefore, there is an α that is an accepting computation of N_T on input $0^j 10^i$.

It remains to show that $|w'| \leq p(|w|)$. To see this, it is sufficient to show that j, i, s and $|\alpha|$ are polynomially bounded in $|w|$. The Gödel-number i is a constant in $|w|$. Parameter j is double-logarithmic, and thus s is polynomially bounded in $|w|$. The computation path α has double-exponential length in i and j and is therefore polynomially bounded in $|w|$. $\square$

4.3 Implications of the Existence of Optimal Proof Systems

In this section, we will see some evidence suggesting optimal proof systems do not exist. One of the implications given by optimal proof systems is the existence of complete sets for NP $\cap$ SPARSE, a consequence which we tend to believe is not true. This result is due to Köbler, Meßner and

Torán. [KMT03]. However our interest in this result goes beyond this evidence, as Buhrman et al. [BFFvM00] showed that there is an oracle such that there are no complete sets for NP $\cap$ SPARSE (see 6.1), although oracles relative to which there are no optimal proof systems have been known even before that [KP89, KMT03].

Theorem 19. *If there is an optimal proof system, then complete sets for* NP $\cap$ SPARSE *exist.*

Proof. We define the set SP, containing descriptions of non-deterministic Turing machines that have runtime bounded by l and accept, up to a given length n, only l different strings:

$$
\begin{aligned}
SP = \{\langle N, 0^l, 0^n \rangle \mid \quad & (1)\ N \text{ is a non-deterministic Turing machine} \\
& (2) \text{ there are at most } l \text{ pairs } (x_i, y_i) \text{ such that} \\
& \qquad \text{(a) all } x_i \text{ are distinct} \\
& \qquad \text{(b) all } y_i \text{ are distinct} \\
& \qquad \text{(c) } |x_i| \leq n, |y_i| \leq l \\
& \qquad \text{(d) } N \text{ accepts } x_i \text{ on path } y_i \}
\end{aligned}
$$

A tuple $\langle N, 0^l, 0^n \rangle$ does not belong to SP if and only if there exist $l+1$ inputs x_i of length at most n that are accepted by N, which proves that $SP \in$ coNP.

We will now define subsets of SP that can be decided in deterministic polynomial time. Assume M is a non-deterministic Turing machine with polynomial runtime q such that for every n, M accepts at most $q(n)$ strings of length at most n. That is, the language accepted by M, $L(M)$ is q-sparse. Observe that the set $SP_M = \{\langle M, 0^{q(n)}, 0^n \rangle \mid n \geq 1\}$ is a subset of SP, as there are at most $l = q(n)$ pairwise different inputs accepted by M for each n (see condition (2)(a) in the definition of SP). For every such M, there is a deterministic polynomial-time Turing machine T_M that decides SP_M: given an input $\langle N, 0^l, 0^n \rangle$, it checks whether $N = M$ and $l = q(n)$, where M and q are coded into T_M's program. We will use SP_M later to show the completeness.

We are going to define the set $S \in$ NP $\cap$ SPARSE, and prove it is complete for that class. The fact that there is an optimal proof system will yield the many-one reduction. So let h be an optimal proof system and let SP reduce to TAUT via γ, which gives us $z \in SP \iff \gamma(z) \in$ TAUT. Then we define

$$
\begin{aligned}
S = \{\langle 0^N, 0^j, x\rangle \mid \quad & \text{(I) } N \text{ is non-det. Turing machine} \\
& \text{(II) there exists } l \text{ and } w,\ |w| \leq j, \\
& \qquad \text{(a) } h(w) = \gamma(\langle N, 0^l, 0^{|x|}\rangle), \\
& \qquad \text{(b) } N \text{ accepts } x \text{ in at most } l \text{ steps.}\}
\end{aligned}
$$

We can see S belongs to NP because of the polynomial-time condition on the tuple. To see S is sparse, first fix an N and j. By condition (II)(b), every x such that $\langle 0^N, 0^j, x\rangle \in S$ is accepted by N in at most l steps. Since $\langle N, 0^l, 0^{|x|}\rangle \in SP$ by (II)(a), we have N only accepting at most l inputs of length at most $|x|$. For the fixed N and j we thus have at most l tuples $\langle 0^N, 0^j, x\rangle \in S$. By condition (II)(a), we can relate this upper bound to the length of the tuples: since h and γ are both polynomial length bounded, l is bounded by some polynomial in j. Therefore exist for any given N and j only a in j polynomial number of tuples in S. By the tally encoding of N and j, there exist only a polynomial number of different N and j for any fixed length k of tuples in S.

Now let's see how every set in NP∩SPARSE many-one reduces to S. Let S' be a set in NP∩SPARSE that is accepted by M in time q. As shown before, SP_M can then be decided in polynomial time. This enables us to define a polynomial-time function

$$
g_M(x) = \begin{cases} \gamma(x) & \text{if } x \in SP_M, \\ \perp & \text{otherwise} \end{cases}
$$

with range TAUT. That is, g_M is a proof system and thus simulated by the optimal proof system h. Hence, there exists a translation function λ and a polynomial r such that for all g_M-proofs x, we have $h(\lambda(x)) = g_M(x)$ and $|\lambda(x)| \leq r(|x|)$. We can thus reduce S' to S via the polynomial-time function $x \mapsto \langle 0^M, 0^{r(|x|)}, x\rangle$.

To prove this claim, assume $x \in S'$. By definition we have $z = \langle M, 0^{q(|x|)}, 0^{|x|}\rangle \in SP_M$. Thus, z is a g_M-proof for $\gamma(z)$, and therefore $\lambda(z)$ is an h-proof for $\gamma(z)$, so $w = \lambda(z)$ satisfies condition (II)(a). Condition (I) of S is fulfilled by definition. For the length bound of (II), notice $|w| = |\lambda(z)| \leq r(|x|) = j$. Since $\lambda(z)$ is an h-proof for $\gamma(z)$, we have $z = \langle N, 0^l, 0^{|x|}\rangle \in SP$. We thus know by definition of SP that N accepts inputs of length at most $|x|$ in at most l steps. This satisfies condition (II)(b). Altogether, we have $\langle 0^M, 0^{r(|x|)}, x\rangle \in S$. The converse follows immediately from (II)(b). $\square$

The technique of this proof can be generalized and extended to a lot of promise classes, most interestingly UP:

Theorem 20. *1. If there is a p-optimal proof system, then* UP *has a many-one complete set.*

2. If there is an optimal proof system, then UP *has a complete set under non-uniform many-one reducibility.*

For the proof, we refer the reader to the work of Köbler, Meßner and Torán [KMT03]. Among UP, it also contains completeness results on Few,FewP, NP $\cap$ SPARSE and NP $\cap$ coNP.

One of the most outstanding consequence of the existence of optimal proof systems is the existence of complete NP-pairs, first proven by Razborov in 1994. The proof requires some preparation and is demonstrated in the next section.

5 Canonical Disjoint NP-pairs for Proof Systems

Razborov found a way to relate proof systems with disjoint NP-pairs [Raz94] by defining a *canonical pair* $(\mathrm{SAT}^*, \mathrm{REF}_f)$ for every proof system f, where

$$\begin{aligned}\mathrm{SAT}^* &= \{(\varphi, 1^m) \mid m \geq 0\},\\ \mathrm{REF}_f &= \{(\varphi, 1^m) \mid \neg\varphi \in \mathrm{TAUT} \text{ and } \exists y, |y| \leq m \text{ such that } f(y) = \neg\varphi\}.\end{aligned}$$

Notice, if $\neg\varphi \in \mathrm{TAUT}$ then φ cannot by satisfied by any assignment, and there exists an f-proof for φ. Hence, REF_f holds pairs $(\varphi, 1^m)$ for all unsatisfiable formulas φ for sufficiently large m. It is thus disjoint from SAT^*, which holds $(\varphi, 1^m)$ only for satisfiable formulas φ. The set REF_f is in NP because f is polynomial-time computable. We can relate REF_f to the question of shortest proofs for tautologies by finding the minimum m for a given tautology $\neg\varphi$.

The notion of canonical pairs is closely related to the notion of simulation of proof systems and yields an corollary originally due to Razborov [Raz94].

Lemma 21. *For two proof systems f and g, if f simulates g, then* $(\mathrm{SAT}^*, \mathrm{REF}_g) \leq_m^{pp} (\mathrm{SAT}^*, \mathrm{REF}_f)$.

Proof. Since f simulates g, there is a function h such that for all strings w, $g(w) = f(h(w))$ and $|h(w)| \leq p(|w|)$. Let $\lambda : \Sigma^* \to \Sigma^*$ be a function mapping $(w, 0^n)$ to $(w, 0^{p(n)})$. We claim that for λ, we have $\lambda(\mathrm{SAT}^*) \subseteq \mathrm{SAT}^*$ and $\lambda(\mathrm{REF}_g) \subseteq \lambda(\mathrm{REF}_f)$. For the first claim, if $(w, 0^n) \in \mathrm{SAT}^*$, then for any $m \in \mathbb{N}$ we have $(w, 0^m) \in \mathrm{SAT}^*$ by definition. For the second claim, if $(w, 0^n) \in \mathrm{REF}_g$, then $\neg w$ is a tautology and there exists a y, $|y| \leq n$, such that $g(y) = \neg w$. Applying h to y yields $\neg w = g(y) = f(h(y))$ and $|h(y)| \leq p(|y|)$ and therefore $(w, 0^{p(n)}) \in \mathrm{REF}_f$. $\square$

With Lemma 21, we can immediately prove the following Corollary.

Corollary 22. *For an optimal proof system f, the pair* $(\mathrm{SAT}^*, \mathrm{REF}_f)$ *is complete for* DisjNP.

This result is an important connection of the theory of proof systems and the theory of disjoint NP-pairs. It gives us insight in more sufficient conditions for the existence of complete pairs. An important open question is whether the converse holds. Does the existence of a many-one complete disjoint NP-pair imply the existence of an optimal proof system? While the answer remains unknown, oracles for both options are known (see Section 6.2).

6 Relativized Worlds

Lacking the ability to prove unrelativized results, a lot of open questions have been studied in detail using oracle Turing machines. This provides some evidence for possible solutions of open problems as well as gives a hint which proof techniques to use to study unresolved problems.

6.1 Existence Optimal and p-Optimal Proof Systems

Fortnow and respectively Meßner and Torán found oracles relative to which there is no optimal respectively no p-optimal proof system. Previously, Meßner and Torán proved [MT97] that the existence of p-optimal proof systems implies the existence of complete sets in UP. They also showed that the weaker assumption of the existence of an optimal proof system is sufficient for the existence of log-space complete set in $\mathrm{NP} \cap \mathrm{SPARSE}$ (see 4.3, in particular Theorem 20 as well as [MT97, KMT03]). We summarize their results as follows.

Proposition 23. 1. *If there is a p-optimal proof system, then* UP *has a many-one complete set.*

2. *If there is an optimal proof system, then complete sets for* $\mathrm{NP} \cap \mathrm{SPARSE}$ *exist.*

Since Hartmanis and Hemachandra exhibited an oracle relative to which UP does not have a many-one complete set [HH88], this immediately gives us an oracle relative to which p-optimal proof systems do not exist. By the results we mentioned earlier for p-optimal proof systems, this also means that relative to this oracle, $\mathrm{E} \neq \mathrm{NE}$ and $\mathrm{P} \neq \mathrm{NP}$.

Buhrman, Fenner, Fortnow and van Melkebeek found an oracle relative to which $\mathrm{NP} \cap \mathrm{SPARSE}$ does not have complete sets. Together with Proposition 23, this gives a relativized world where optimal proof systems do not exist.

Glaßer, Selman, Sengupta and Zhang [GSSZ03, Chapter 6] construct an oracle O_1 relative to which $\mathrm{NE} = \mathrm{coNE}$ and therefore, by Theorem 18, optimal proof systems do exist. This and the oracle O_2 from the same paper are also interesting for the next section.

6.2 Converse of Razborov

The oracles O_1 and O_2 by Glaßer, Selman, Sengupta and Zhang [GSSZ03, Chapter 6] provide insight into the question of whether the converse of Razborov's Theorem holds. That is, does the existence of a complete pair in DisjNP imply the existence of an optimal proof system? The question remains open, but Glaßer et al. proved that it cannot be answered with a relativizable proof. In particular, for both O_1 and O_2 complete pairs exist, but optimal proof systems exists only for O_1. For O_2, there are no optimal proof systems. It is also worth to mention that relative to both oracles, the ESY-conjectures holds.

6.3 Separation of ESY refinements

Glaßer and Wechsung constructed an oracle D relative to which UP = NP and NP $\neq$ coNP [GW03]. Along with the results we know about ESY-tt, which does not hold if UP = NP (see Theorem 16), and Theorem 15, where we prove that NP $\neq$ coNP is equivalent to ESY-m, this oracle separates ESY-tt from ESY-m. Notice that therefore, relative to D, ESY does not hold, but ESY-m does.

7 Conclusion

7.1 Open Questions and Future Work

1. Oracle for which converse of Razborov holds
2. Oracle such that there is a optimal, but not a p-optimal proof system

References

[BFFvM00] H. Buhrman, S. A. Fenner, L. Fortnow, and D. van Melkebeek. Optimal proof systems and sparse sets. In Horst Reichel and Sophie Tison, editors, *STACS*, volume 1770 of *Lecture Notes in Computer Science*, pages 407–418. Springer, 2000.

[CR79] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[ESY84] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[Gol05] Oded Goldreich. On promise problems (a survey in memory of shimon even [1935-2004]). *Electronic Colloquium on Computational Complexity (ECCC)*, (018), 2005.

[GS88] J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.

[GSSZ03] C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(011), 2003.

[GW03] Christian Glaßer and Gerd Wechsung. Relativizing function classes. *J. UCS*, 9(1):34–50, 2003.

[HH88] J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theor. Comput. Sci.*, 58(1-3):129–142, June 1988.

[HPRS12] A. Hughes, A. Pavan, N. Russell, and A. L. Selman. A thirty year old conjecture about promise problems. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 473–484. Springer, 2012.

[KMT03] J. Köbler, J. Meßner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Inf. Comput.*, 184(1):71–92, 2003.

[KP89] J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *J. Symb. Log.*, 54(3):1063–1079, 1989.

[MT97] J. Meßner and J. Torán. Optimal proof systems for propositional logic and complete sets. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(26), 1997.

[Raz94] A. A. Razborov. On provably disjoint NP-pairs. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(6), 1994.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.