Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik IV
Theoretische Informatik

Bachelor Thesis

simulation of proof systems

Nils Wisiol

submitted on May 11, 2012

supervisor:
Dr. Christian Glaßer

Contents

1 Introduction

After Cook stated 1971 the P versus NP question [Coo71], he gave the main motivation to study proof systems in 1979. Cook and Reckhow showed in their article the close relation between the separation of complexity classes and the existence of polynomially bounded proof systems [CR79].

Despite its importance, the P versus NP question remains still open. Informally speaking, P = NP means that for every problem that has a efficiently verifiable solution, we can find that solution efficiently as well. While most theoreticians assume that P $\neq$ NP, their equality would have heavy implications. One consequence affects the security of communication in the Internet, as public-key cryptography depends an the existence of certain problems in NP that are not efficiently to decide. If there are no problems with solutions fast to verify but hard to find, as P = NP would imply, the small lock beside the URL in one's browser could not indicate a secure communication anymore [For09].

P = NP would also have fundamental implications on mathematics: As mathematical proofs must by definition be efficiently verifiable, P = NP would imply that they are efficiently to find [CR79]. This argument to believe that P $\neq$ NP is further illustrated in the following quotation, taken from Aaronsons Blog[1].

> "If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in 'creative leaps', no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss; everyone who could recognize a good investment strategy would be Warren Buffett."

Closely related to the P versus NP problem is the question if NP = co-NP. If P = NP, then NP = co-NP, since P is closed under complement. In return, if one can separate NP from co-NP, then P $\neq$ NP. To connect the field of proof systems with the P versus NP questions, we state

Proposition 1 ([KMT03], [CR79]). *NP = co-NP if and only if a polynomially bounded proof systems for TAUT exists.*

Which we will prove in chapter 3.

To get insight into the field of proof systems, we will first define the notions used in the following chapter. Subsequently, we will give an overview of important results about proof systems in chapter 3. In this chapter, we will also proof proposition 1. After this, we will show the existence of languages without optimal proof systems in certain complexity classes. Finally, we will give a conclusion and look forward to currently unresolved problems and further questions.

this paragraph should be much longer

[1] http://www.scottaaronson.com/blog/

2 Preliminaries

As mentioned before, we will first introduce important symbols and definitions. Although some familiarity with standard notions of complexity theory is assumed, we will here define most of the notions used in this thesis. For the most important ones, we will give a short discussion.

Let $\Sigma = \{0, 1\}$ denote the alphabet. The output of a Turing transducer M on input $x \in \Sigma^*$ is denoted by $M(x)$. If the transducer M does not accept or runs forever on input x, we define $M(x) = \perp$. We say a Turing transducer *calculates* a partial function f, if $M(x) = f(x)$ for all $x \in \Sigma^*$. We further define $\mathrm{time}_M(x)$ as the number of steps the transducer M runs on input $x \in \Sigma^*$. Similar, for a partial function f, we define $\mathrm{time}_f(x) = \mathrm{time}_M(x)$ for a transducer M calculating f. With $\mathcal{FP}$ we denote the set of all partial functions f with $\mathrm{time}_f(x) \leq p(|x|)$ for a polynomial p.

is that well-defined?

Definition 1. *A function $f \in \mathcal{FP}$ is called* proof system *for a language L if the range of f is L. A string w with $h(w) = x$ is called an h-*proof *for x. f is called an* polynomially bounded *proof system, if there is a polynomial p such that for every $x \in L$, there is a h-proof w with $|w| \leq q(|x|)$.*

With this definition, a proof system for L is basically a polynomial-time bounded function that enumerates L. Notice, although its time bound against the input, the shortest proof a string $w \in L$ can be be very long. To give an example, let h be defined by

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

Then h is a proof system for SAT. It is an open question, whether *sat* is p-optimal. Köbler and Messner showed, that this question is equivalent to a variety of well studied complexity theoretic assumptions [KM00]. We will cite some of their results in lemma/chapter .

well, where?

There may be various proof systems for a language L. In order to make them comparable, we define the notion of *simulation* of proof systems. It turns out that the notion of simulation corresponds in a certain way with the notion of many-one reducibility [KM00]. In the following definitions, we will keep this correspondence.

Definition 2. *Let h and h' be proof systems for a language L. If there is a polynomial p and a function f such that for all $w \in \Sigma^*$*

$$h(f(w)) = h'(w)$$

and $|f(w)| \leq p(|w|)$, then h simulates *h'.*

Speaking informally, f translates h-proofs into h'-roofs and keeps them polynomial length bounded. In the given definition, f could be hard or even impossible to calculate. Hence we define a stronger version of this notion, demanding $f \in \mathcal{FP}$.

Definition 3. *Again, let h and h' be proof systems for a language L. If h simulates h' with a function f and additionally $f \in \mathcal{FP}$, h* p-simulates *h'.*

Notice, if f is a function that can be calculated in polynomial time p, then we obtain $|f(w)| \leq p(|w|)$ as required in the definition of simulation. With a proof system p-simulating another, we can translate proofs as above in polynomial short time. With the notion of simulation of proof systems, we can compare different proof systems for a language L. With respect to these notions, we will define a notion of the best proof system as follows.

Definition 4. *A proof system h for L is called* optimal*, if it simulates every proof system for L. It is called* p-optimal*, if it p-simulates every proof system for L.*

Like noted above, this definition corresponds with the definition of complete problems in respect of many-one-reducibility. We will investigate further on the connection between complete problems and optimal proof systems later in lemma/chapter **??** [KMT03].

The existence of optimal proof systems for a arbitrary language L is an important complexity theoretic question. For languages in P and NP, there is always a optimal proof system, as we will see in 3. For super-polynomial time complexity classes, there are languages without an optimal proof system, as we will show in chapter 4. For that reason, we will define a complexity class containing all languages possessing an optimal proof system.

Definition 5. *Let* OPT *be the complexity class of all languages that have a optimal proof system.*

Observe that for OPT we use the weaker notion of simulation. As noted above, we can easily state a proof system for languages in P or NP, therefore we obtain NP $\subseteq$ OPT. It is an open questions whether OPT $\subseteq$ NP.

is there a simulation notion?

With these notions, we will take a look at important results in the field of optimal proof systems in the next chapter. For notions not defined in this thesis, refer to a standard work of computational complexity like the one from Papadimitriou [Pap94].

citation, consequences of OPT $\subseteq$ NP, more info

3 A brief Overview of Proof Systems

After defining the important notions for this thesis, we will give a brief overview of some important results in the field of optimal proof systems.

One basic lemma that is widely used formalizes a part of the connection between optimal proof systems and polynomial many-one-reducibility. Later in this thesis, we will use it to proof corollary 9. The following proof is mainly taken from Köbler et al. [KMT03].

Lemma 2. *If A has a (p-)optimal proof system and if $B \leq_m^p A$, then B has a (p-)optimal proof system, too.*

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

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

is a proof system for B, as $h(w) = f(x) \in A$ is equivalent to $x \in B$. To show h' is optimal, let g' be a proof system for B. In order to obtain a proof system for A, let g be

$$g(bw) = \begin{cases} h(w) & (b = 0), \\ f(g'(w)) & (b = 1). \end{cases}$$

Since both $h(w)$ and $f(g'(w))$ are in A and h is a proof system for A, g is also a proof system for A. As h is p-optimal, there is a function $t \in \mathcal{FP}$ translating g-proofs to h-proofs implying that

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

This implies $h'(\langle g'(w), t(1w) \rangle) = g'(w)$. Hence, h' p-simulates g'. □

In contraposition to this, we can state that for $B \leq_m^p A$, if B has no (p-)optimal proof system, then A has not either.

tell some history of this question [KM00, p. 1]

The following lemma gives a partial answer to the basic question what languages do have optimal proof systems.

Lemma 3. *(i) Every language in P has a p-optimal proof system.*

(ii) Every language in NP has an optimal proof system.

Proof. (i) Let $L \in \mathrm{P}$. Then there is a function $f \in \mathcal{FP}$ with $f(w) = 1 \Leftrightarrow w \in L$. To show there is a proof system, let h be defined by

$$h(w) = \begin{cases} w & (f(w) = 1), \\ \perp & (\text{otherwise}). \end{cases}$$

Then h is a proof system for L. To show h is optimal, let h' be an arbitrary proof system for L. Then $h' \in \mathcal{FP}$ by definition and we can translate h'-proofs with h' in polynomial time into h-proofs, in formulas

$$h(h'(w)) = h'(w).$$

Therefore, h p-simulates every proof system h'.

(ii) Let $L \in \mathrm{NP}$. Then there is a nondeterministic Turing transducer M deciding L in polynomial time. Let $f_i(x) \in \mathcal{FP}$ the function calculating the i-th path of the nondeterministic calculation of M. Finally, let h be defined by

$$h(\langle i, w\rangle) = \begin{cases} f_i(w) & (f_i(w) \text{ accepts}), \\ \perp & (\text{otherwise}). \end{cases}$$

Then $h \in \mathcal{FP}$ is a proof system for L. To show h is optimal, let h' be an arbitrary proof system for L. Let g be a function that maps an $w \in L$ to an i such that $f_i(w)$ accepts in polynomial time. Notice that g may be not in $\mathcal{FP}$. With these definitions, we obtain

$$h(\langle g(h'(w)), h'(w)\rangle) = h'(w).$$

Therefore, h simulates every proof system h' via the translating function

$$w \mapsto \langle g(h'(w)), h'(w)\rangle.$$

proof for "optimal" is missing

□

This implies NP ⊆ OPT. By stating different properties for P and NP, the lemma connects to the P-NP-question. If one would find a set in NP without an p-optimal proof system, one would have separated P from NP.

Corollary 4. *If there is no p-optimal proof system for SAT, then $P \neq NP$.*

In order to prove proposition 1, we will show two lemma of which the first is taken from the work of Cook and Reckhow [CR79]. It gives an equivalent formulation of the question whether NP = co-NP.

Lemma 5. *$NP =$ co-NP if and only if $TAUT \in NP$.*

Proof. Without further assumptions, we can show by using a nondeterministic Turing transducer, that an arbitrary formula is not a tautology in polynomial time by guessing and verifying an assignment for which the formula is falsified. Thus, $\overline{\mathrm{TAUT}} \in \mathrm{NP}$.

Assume $\mathrm{TAUT} \in \mathrm{NP}$, and let L be an arbitrary language with $L \in \mathrm{NP}$. Hence, there is a function $f \in \mathcal{FP}$ such that $x \in L \Leftrightarrow f(x) \in \overline{\mathrm{TAUT}}$ respectively $x \in \overline{L} \Leftrightarrow f(x) \in \mathrm{TAUT}$. Since $\mathrm{TAUT} \in \mathrm{NP}$, for any given $x \in \Sigma^*$ we can in nondeterministic polynomial time decide whether $f(x)$ is in TAUT. Thus we also can decide in nondeterministic polynomial time whether $x \in \overline{L}$, as $f \in \mathcal{FP}$. It follows that $\overline{L} \in \mathrm{NP}$. As this proves that NP is closed under complement, we obtain NP ⊆ co-NP. So see that co-NP ⊆ NP, let an arbitrary language $\overline{L}$ be in co-NP. By definition we obtain $L \in \mathrm{NP}$. As NP is closed under complement, $\overline{L} \in \mathrm{NP}$. Thus, co-NP ⊆ NP.

insert part of the proof of theorem 1 from [Coo71].

Assume $\mathrm{TAUT} \notin \mathrm{NP}$, then $\overline{\mathrm{TAUT}} \notin$ co-NP. As we have seen above, $\overline{\mathrm{TAUT}} \in \mathrm{NP}$. Hence NP ≠ co-NP. □

The second lemma connects with the theory of proof systems by formulating a necessary and sufficient condition for a language being in NP [CR79].

Lemma 6. *A set $L \neq \emptyset$ is in NP if and only if L has a polynomially bounded proof system.*

Proof. Assume $L \in \mathrm{NP}$, then some nondeterministic Turing transducer M accepts L in polynomial time. Let $f_i(x) \in \mathcal{FP}$ the function calculating the i-th path of the nondeterministic calculation of M. We define f by

$$f(\langle i, x\rangle) = \begin{cases} x & (f_i(x) \text{ accepts}), \\ \perp & (\text{otherwise}). \end{cases}$$

Then f is a polynomially bounded proof system for L.

Conversely, assume f is a polynomially bounded proof system for L. Then a nondeterministic Turing transducer on input y can guess a proof x and verify $f(x) = y$. $\square$

Putting lemma 5 and 6 together, we obtain proposition 1,

$$\text{NP} = \text{co-NP} \Leftrightarrow \text{ there is a polynomially bounded proof system for TAUT.}$$

Using this theorem, one tried to seperate NP from co-NP by studying more and more powerful proof systems, showing that they are not polynomially bounded [KMT03]. As mentioned before, there was no success on answering this questions until now. To take the notion of optimal proof systems into account, one could ask if there is an optimal or even p-optimal proof system for TAUT. If that were the case, then the existence of one specific proof system that is not polynomially bounded would suffice to proof that NP $\neq$ co-NP and hence P $\neq$ NP [KMT03].

Krajícek and Pudlák proved a sufficient condition for the existence of optimal proof systems for TAUT [KP89].

Theorem 7. *If NE = co-NE then optimal proof systems for TAUT exist. If E = NE then p-optimal proof systems for TAUT exists.*

We will omit their proof in this thesis, since it uses many notions of formal logics and a huge equivalence theorem not introduced here.

With these basic results, we will investigate the question whether there are languages possessing optimal proof systems outside of NP.

4 A set in co-NEXP \ OPT

A first step in our investigation whether OPT $\subseteq$ NP is to show that for super-polynomial complexity classes there are at least some languages not possessing a optimal proof system.

Theorem 8. *Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function such that for every polynomial p there is a number n with $p(n) \leq t(n)$. Then there is a language $L \in$ co-NTIME$(t(n))$ that has no optimal proof system.*

Messner showed that under the same presumptions as in our theorem, there is a language $L \in$ co-NTIME$(t(n))$ without an optimal acceptor [Mes99]. He also proofed that the existence of a optimal acceptor is equivalent to the existence of a optimal proof system for every p-cylinder L. We will here give a proof that is based on the work of Messner, but stays in the notion of proof systems.

Proof. Let $f_1, f_2, ...$ be a enumeration of all $\mathcal{FP}$-functions with time$(f_i) \leq n^i + i$. For any $i > 0$, let L_i be the regular language described by the expression 0^i10^*. Define

How is that obtained?

$$L'_i = \{x \in L_i | \forall_{y \in \Sigma^*} |y|^{2i} \leq t(|x|) \implies f_i(y) \neq x\}.$$

That is, as long as you put strings of length $|y|^{2i} \leq t(|x|)$ into f_i, you will not obtain x. Let $L = \bigcup_{i>0} L'_i$.

First, we obtain $L \in$ co-NTIME$(t(n))$. To show this, one considers

$$L \in \text{co-NTIME} \Leftrightarrow \overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i} \in \text{NTIME}.$$

By negating the condition for L'_i, we get

$$\overline{L'_i} = \{x \in \Sigma^* | x \notin L_i \vee (\exists_{y \in \Sigma^*} |y| \leq t(|x|) \wedge f_i(y) = x)\}.$$

For any given x, we can decide in polynomial time whether it is in any L_i or not. If it is not, then x is in $\overline{L'_i}$ for all $i > 0$ and therefore $x \in \overline{L}$, so we are done. If it is in any L_i, it is in exactly one L_i. Let i^* be the set with $x \in L_{i^*}$. We can simulate a polynomial-time machine calculating $f_{i^*}(y)$ on every input $y \in \Sigma^*$ with $|y|^{2i} \leq t(|x|)$. If, and only if, there is a path with $f_{i^*}(y) = x$, then $x \in \overline{L}$. In both cases, $\overline{L} \in$ NTIME$(t(n))$.

have a look at simulation runtime

For a proof system f_i with $f_i(\Sigma^*) = L$, we observe that $L'_i = L_i$. Assume there is an $x = 0^i1z \in L_i$ that is not in L'_i. Then there is an y with $|y|^{2i} \leq t(|x|)$ and $f_i(y) = x$. Since f_i is a proof system for L, this yields $x = 0^i1z \in L$ and so $x \in L'_i$, which contradicts the assumption. Therefore, for any y with $f_i(y) = x \in L_i$ we know that $|y|^{2i} > t(|x|)$. Speaking informally, every proof system f_i for L has "long" proofs on $L'_i \subset L$.

Assume now, for contradiction, that f_i is a optimal proof system for L. Let g be a function defined as

$$g(bx) = \begin{cases} f_i(x) & (b = 0), \\ x & (b = 1 \text{ and } x = 0^i10^* \in L_i = L'_i). \end{cases}$$

g is a proof system for L with polynomial length-bounded proofs for all $x \in L_i$. As f_i is optimal, there is a function f^* such that for all $x \in L'_i$, $f_i(f^*(x)) = g(x)$ and $|f^*(x)| \leq p(|x|)$ for a polynomial p. Let q be the polynomial $q(n) = p(n)^{2i}$. As $p(|x|)$ is positive, $p(|x|) \leq$

$p(|x|)^{2i}$. As there is an n with $q(n) \leq t(n)$, there is an x in L_i such that $|f^*(x)| \leq p(|x|) \leq q(|x|) = p(|x|)^{2i} \leq t(|x|)$. According to the definition of L'_i, this yields $f_i(f^*(x)) \neq x$. Therefore, f_i is not optimal on L'_i, which contradicts the assumption that f_i is optimal on L. □

Now, let us take a closer look at this set L that has no optimal proof system. One first observation is that L is sparse. As every L'_i only contains strings that are of the form 0^i10^*, L is a subset of the regular language $L_R = 0^*10^*$. Therefore, the density of L_R is an upper bound for the density of L. As $\text{dens}_{L_R}(n) = n$, L_R and L are both sparse.

Using the relation to many-one-hard reducible sets proofed in lemma 2, we obtain

Corollary 9. *No set $\leq_m^p$-hard for co-NE has an optimal proof system.*

Is this possible for co-NEXP?

Proof. With $t(n) = 2^n$, we can get an $L \in$ co-NE that has no optimal proof system. Any $\leq_m^p$-hard set A for co-NE is $L \leq_m^p A$. Together with the cited result we obtain, that A cannot have optimal proof system. □

5 Conclusion and future work

What a great work!

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen Hilfsmittel und Quellen als die angegebenen benutzt habe. Weiterhin versichere ich, die Arbeit weder bisher noch gleichzeitig einer anderen Prüfungsbehörde vorgelegt zu haben.

Würzburg, den ____________________, __________________________________

(Nils Wisiol)

Bibliography

[Coo71] Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing (New York, NY, USA), STOC '71, ACM, 1971, pp. 151–158.

[CR79] Stephen A. Cook and Robert A. Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic **44** (1979), 36–50.

[For09] Lance Fortnow, *The status of the p versus np problem*, Commun. ACM **52** (2009), no. 9, 78–86.

[KM00] Johannes Köbler and Jochen Messner, *Is the standard proof system for sat p-optimal?*, Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (London, UK, UK), FST TCS 2000, Springer-Verlag, 2000, pp. 361–372.

[KMT03] Johannes Köbler, Jochen Messner, and Jacobo Torán, *Optimal proof systems imply complete sets for promise classes*, Inf. Comput. **184** (2003), no. 1, 71–92.

[KP89] Jan Krajícek and Pavel Pudlák, *Propositional proof systems, the consistency of first order theories and the complexity of computations*, J. Symb. Log. **54** (1989), no. 3, 1063–1079.

[Mes99] Jochen Messner, *On optimal algorithms and optimal proof systems*, Proceedings of the 16th annual conference on Theoretical aspects of computer science (Berlin, Heidelberg), STACS'99, Springer-Verlag, 1999, pp. 541–550.

[Pap94] Christos H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994.