Electronic Colloquium on Computational Complexity, Report No. 92 (2009)

Proof Systems that Take Advice[⋆]

Olaf Beyersdorff[1], Johannes Köbler[2], and Sebastian Müller[2]

[1] Institut für Theoretische Informatik, Leibniz-Universität Hannover, Germany
beyersdorff@thi.uni-hannover.de
[2] Institut für Informatik, Humboldt-Universität zu Berlin, Germany
{koebler,smueller}@informatik.hu-berlin.de

Abstract. One of the starting points of propositional proof complexity is the seminal paper by Cook and Reckhow [CR79], where they defined propositional proof systems as poly-time computable functions which have all propositional tautologies as their range. Motivated by provability consequences in bounded arithmetic, Cook and Krajíček [CK07] have recently started the investigation of proof systems which are computed by poly-time functions using advice.
In this paper we concentrate on three fundamental questions regarding this new model. First, we investigate whether a given a language L admits a polynomially bounded proof system with advice. Depending on the complexity of the underlying language L and the amount and type of the advice used by the proof system, we obtain different characterizations for this problem. In particular, we show that this question is tightly linked with the question whether L has small nondeterministic instance complexity.
The second question concerns the existence of optimal proof systems with advice. For propositional proof systems, Cook and Krajíček [CK07] gave a surprising positive answer which we extend to all languages.
These results show that using advice yields a more powerful model, but it is also less directly applicable in practice. Our third question therefore asks whether the usage of advice in propositional proof systems can be simplified or even eliminated. While in principle, the advice can be very complex, we show that propositional proof systems with logarithmic advice are also computable in poly-time with access to a sparse NP-oracle. Employing a recent technique of Buhrman and Hitchcock [BH08] we also manage to transfer the advice from the proof to the formula, which leads to an easier computational model.

1 Introduction

Propositional proof complexity studies the question how difficult it is to prove propositional tautologies. In the classical Cook-Reckhow model, proofs are verified in deterministic polynomial time [CR79]. While this is certainly the most useful setting for practical applications, it is nevertheless interesting to ask if proofs can be shortened when their verification is possible with stronger computational resources. In this direction, Cook and Krajíček [CK07] have recently initiated the study of proof systems which use advice for the verification of proofs. Their results show that, like in the classical Cook-Reckhow setting, these proof systems enjoy a close connection to theories of bounded arithmetic.

In this paper we continue their investigation and particularly focus on the following fundamental questions for this new model:

[⋆] Preliminary versions of the results in this paper have appeared in the proceedings of LATA'09 [BKM09] and SAT'09 [BM09]. Part of the results of this paper are also contained in the technical report TR08-075 at ECCC. Research was supported by DFG grant KO 1053/5-2.

ISSN 1433-8092

Q1: Given a language L, do there exist polynomially bounded
proof systems with advice for L?
Q2: Do there exist optimal proof systems with advice for L?
Q3: For propositional proof systems, does advice help to shorten proofs?

For question Q1, one of the major motivations for proof complexity [CR79], we obtain a complete complexity-theoretic characterization. The classical Cook-Reckhow Theorem states that $\mathsf{NP} = \mathsf{coNP}$ if and only if the set of all tautologies TAUT has a polynomially bounded proof system, i.e., there exists a polynomial p such that every tautology φ has a proof of size $\leq p(|\varphi|)$ in the system. Consequently, showing super-polynomial lower bounds to the proof size in propositional proof systems of increasing strength provides one way to attack the P/NP problem. This approach, also known as the Cook-Reckhow program, has lead to a very fruitful research on the length of propositional proofs (cf. [Pud98]).

As in the Cook-Reckhow Theorem above, we obtain a series of results leading to a complete characterization for Q1. In particular, we show a tight connection of this problem to the notion of nondeterministic instance complexity. Similarly as Kolmogorov complexity, instance complexity measures the complexity of individual instances of a language [OKSW94]. In its nondeterministic version, Arvind, Köbler, Mundhenk, and Torán [AKMT00] used this complexity measure to show that, under reasonable complexity-theoretic assumptions, there are infinitely many tautologies that are hard to prove in every propositional proof system. In the light of our present contribution, this connection between nondeterministic instance complexity and proof complexity is strengthened by results of the following form: *all elements of a given language L have small instance complexity if and only if L has a proof system with advice such that every $x \in L$ has a short proof.*

While the existence of optimal proof systems in the classical model is a prominent open problem posed by Krajíček and Pudlák twenty years ago [KP89], question Q2 receives a surprising positive answer: optimal proof systems exist when a small amount of advice is allowed. For propositional proof systems this was already shown by Cook and Krajíček [CK07]. Using the proof technique from [CK07], we show that for every language L, the class of all proof systems for L using logarithmic advice contains an optimal proof system.

For question Q3 we concentrate on the most interesting case of propositional proof systems. Unfortunately, proof systems with advice do not constitute a feasible model for the verification of proofs in practice, as the non-uniform advice can be very complex (and even non-recursive). Approaching question Q3, we therefore investigate whether the advice can be simplified or even eliminated while still preserving the same upper bounds on the lengths of proofs. Our first result in this direction shows that proving propositional tautologies does not require complicated or even non-recursive advice: every propositional proof system with up to logarithmic advice is simulated by a propositional proof system computable in polynomial time with access to a sparse NP-oracle. Thus in propositional proof complexity, computation with advice can be replaced by a more realistic computational model.

While this result holds unconditionally, our next two results explore consequences of a positive or negative answer to question Q3. Assume first that advice helps to prove tautologies in the sense that proof systems with advice admit non-trivial upper bounds on the lengths of proofs. Then we show that the same upper bound can be achieved in a proof system with a simplified advice model. On the other hand, if the answer is negative in the sense that advice does not help to shorten proofs even for simple tautologies, then we obtain optimal propositional proof systems without advice.

This paper is organized as follows. After reviewing some facts from computational complexity in Sect. 2, we start in Sect. 3 by introducing our general model for proof systems with advice. In Sect. 4 we show that in the propositional case, proof systems with logarithmic advice are simulated by proof systems computable in poly-time with access to a sparse NP-oracle.

Section 5 contains our results on optimal proof systems with advice (Q2). Before we turn to question Q1, we review the notion of instance complexity and related complexity classes in Sect. 6. In particular, we prove strict inclusions for a chain of complexity classes which play a central role in our characterization of Q1. In Sect. 7 we investigate Q1 for arbitrary languages, whereas in Sect. 8 we focus on TAUT which presents the most interesting case for practical applications.

Again, in Sect. 9 we concentrate on propositional proof systems and contribute to an answer to Q3 by proving that advice can be transferred from the proof to the formula, leading to an easier computational model. We obtain this result by employing a recent technique by Buhrman and Hitchcock [BH08]. Finally, in Sect. 10 we conclude with a discussion and some directions to future research.

2 Preliminaries

We assume familiarity with standard complexity classes (cf. [BDG88]). In the following we just mention a few classes which occur in this paper. The *Boolean hierarchy* BH is the closure of NP under union, intersection, and complementation. The levels of BH are denoted BH_k, where BH_2 is also known as $\mathsf{D^p}$. The Boolean hierarchy coincides with $\mathsf{P}^{\mathsf{NP}[O(1)]}$ consisting of all languages which can be solved in polynomial time with constantly many queries to an NP oracle. If we allow $O(\log n)$ adaptive queries we get the presumably larger class $\mathsf{P}^{\mathsf{NP}[\log]}$.

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 $n \in \mathbb{N}$, $|A \cap \Sigma^n| \le p(n)$. 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.

Complexity classes with *advice* were first considered by Karp and Lipton [KL80]. For each function $h : \mathbb{N} \to \Sigma^*$ and each language L we let $L/h = \{x \mid \langle x, h(|x|)\rangle \in L\}$. If C is a complexity class and F is a class of functions, then $\mathsf{C}/F = \{L/h \mid L \in \mathsf{C}, h \in F\}$. Usually the family of functions F is defined by some bound on the length of the values in terms of the argument. Thus, for example, $\mathsf{NP}/O(1)$ denotes the class of languages recognized by NP machines with advice functions h where $|h(n)|$ is bounded by a constant (cf. [BDG88]).

3 Proof Systems with Advice

We start with a general semantic definition of proof systems:

Definition 1. *A* proof system *for a language L is a (possibly partial) surjective function $f : \Sigma^* \to L$. For $L = \mathrm{TAUT}$, f is called a* propositional proof system.

A string w with $f(w) = x$ is called an *f-proof* of x. Proof complexity studies lengths of proofs, so we use the following notion: for a function $t : \mathbb{N} \to \mathbb{N}$, a proof system f for L is *t-bounded* if every $x \in L$ has an f-proof of size $\leq t(|x|)$. If t is a polynomial, then f is called *polynomially bounded*.

Proof systems are compared according to their strength by simulations as introduced in [CR79] and [KP89]. If f and g are proof systems for L, we say that g *simulates* f (denoted $f \leq g$), if there exists a polynomial p such that for all $x \in L$ and f-proofs w of x there is a g-proof w' of x with $|w'| \leq p(|w|)$. If such a proof w' can even be computed from w in polynomial time, we say that g *p-simulates* f and denote this by $f \leq_p g$. If the systems f and g mutually (p-)simulate each other they are called *(p-)equivalent*.

In the classical framework of Cook and Reckhow [CR79], proof systems are additionally required to be computable in polynomial time. Recently, Cook and Krajíček [CK07] have started to investigate propositional proof systems that are computable in polynomial time with the help of advice. We will first generalize this concept to arbitrary languages.

Our general model of computation for proof systems f with advice is a polynomial-time Turing transducer with several tapes: an input tape containing the proof π, possibly several work tapes for the computation of the machine, an output tape where we output the proven element $f(\pi)$, and an advice tape containing the advice. We start with a quite flexible definition of proof systems with advice for arbitrary languages, generalizing the notion of propositional proof systems with advice from [CK07] and [BM].

Definition 2. *For a function $k : \mathbb{N} \to \mathbb{N}$, a proof system f for L is a* proof system with k bits of advice, *if there exist a polynomial-time Turing transducer M, an advice function $h : \mathbb{N} \to \Sigma^*$, and an advice selector function $\ell : \Sigma^* \to 1^*$ such that*

1. *ℓ is computable in polynomial time,*
2. *M computes the proof system f with the help of the advice h, i.e., for all $\pi \in \Sigma^*$, $f(\pi) = M(\pi, h(|\ell(\pi)|))$, and*
3. *for all $n \in \mathbb{N}$, the length of the advice $h(n)$ is bounded by $k(n)$.*

For a class F of functions, we denote by ps/F the class of all ps/k with $k \in F$.

We say that f *uses k bits of input advice* if ℓ has the special form $\ell(\pi) = 1^{|\pi|}$. On the other hand, in case $\ell(\pi) = 1^{|f(\pi)|}$ for all π in the domain of f, then f is said to *use k bits of output advice.* By this definition, proof systems with input advice use non-uniform information depending on the length of the proof, while proof systems with output advice use non-uniform information depending on the length of the proven formula.

We note that proof systems with advice are a quite powerful concept, as for every language $L \subseteq \Sigma^*$ there exists a proof system for L with only one bit of advice. In contrast, the class of all languages for which proof systems without advice exist coincides with the class of all recursively enumerable languages.

The above definition of a proof system with advice allows a very liberal use of advice, in the sense that for each input, the advice string used is determined by the advice selector function ℓ. For $L = \mathrm{TAUT}$ this general definition coincides with our definition of propositional proof systems with advice from [BM]. In [CK07] and [BM], concrete proof systems arising from extensions of *EF* were investigated, which indeed require this general framework with respect to the advice.

In the next proposition we observe that proof systems with input advice are already as powerful as our general model of proof systems with advice.

Proposition 3. *Let $k : \mathbb{N} \to \mathbb{N}$ be a monotone function, $L \subseteq \Sigma^*$, and f be a ps/k for L. Then there exists a proof system f' for L with k bits of input advice such that f and f' are p-equivalent.*

Proof. We choose a polynomial-time computable bijective pairing function $\langle \cdot, \cdot \rangle$ on $\mathbb{N}$ such that $\langle n_1, n_2 \rangle \geq n_1 + n_2$ for all numbers n_1 and n_2. Let f be a ps/k for L with advice function h and advice selector ℓ. We define a proof system f' for L with input advice as follows: on input π' of length n the function f' first computes the two unique numbers n_1 and n_2 such that $n = \langle n_1, n_2 \rangle$. It then interprets the first n_1 bits $\pi'_1 \dots \pi'_{n_1}$ of π' as an f-proof π and checks whether $\ell(\pi) = 1^{n_2}$. If this is the case, $f'(\pi') = f(\pi)$, otherwise f' outputs a fixed element $x_0 \in L$. Obviously, $f'(\pi')$ is computable with advice $h(|\ell(\pi)|) = h(n_2)$ whose length is bounded by $k(n_1) \leq k(n)$. This shows that f' is a ps/k for L with input advice.

The p-simulation of f by f' is computed by the function $\pi \mapsto \pi' = \pi 1^m$ where $m = \langle |\pi|, |\ell(\pi)| \rangle - |\pi|$. The converse simulation $f' \leq_p f$ is given by

$$\pi' \mapsto \begin{cases} \pi = \pi'_1 \dots \pi'_{n_1} & \text{if } |\pi'| = \langle n_1, n_2 \rangle \text{ and } \ell(\pi) = 1^{n_2} \\ \pi_0 & \text{otherwise,} \end{cases}$$

where π_0 is a fixed f-proof of x_0. □

4 Substituting Advice by Weak Oracles

From a practical point of view, proof systems with advice are susceptive to criticism: advice can be arbitrarily complex (even non-recursive) and thus computing proofs with advice does not form a feasible model to use in practice. Our next result shows that instead of possibly complex non-uniform information we can also use sparse NP-oracles to achieve the same proof lengths as in propositional proof systems with advice.

Theorem 4.

1. *Every propositional proof system with logarithmic advice is simulated by a propositional proof system computable in polynomial time with access to a sparse* NP*-oracle.*

2. *Conversely, every propositional proof system computable in polynomial time with access to a sparse* NP*-oracle is simulated by a propositional proof system with logarithmic advice.*

Proof. For the first claim, let f be a propositional proof system computed by the polynomial-time Turing transducer M_f with advice function h_f where $|h_f(n)| \leq c \cdot \log n$ for some constant c. Without loss of generality, we may assume that f uses input advice (Proposition 3). We choose a length-injective polynomial-time computable pairing function $\langle \cdot \rangle$ and consider the set

$$A = \Big\{ \langle 1^n, a \rangle \mid a \in \Sigma^{\leq c \cdot \log n} \text{ and for some } \pi \in \Sigma^n,\ M_f(\pi, a) \notin \mathrm{TAUT} \Big\} \ ,$$

where $M_f(\pi, a)$ denotes the computation of M_f on input π under advice a. Intuitively, A collects all incorrect advice strings for M_f on length n. By construction, A is sparse. Further, $A \in \mathsf{NP}$ because on input $\langle 1^n, a \rangle$ we can guess $\pi \in \Sigma^n$ and nondeterministically verify $M_f(\pi, a) \notin \mathrm{TAUT}$ by guessing a satisfying assignment for $\neg M_f(\pi, a)$.

We now construct a polynomial-time oracle Turing transducer M_g which under oracle A computes a proof system $g \geq f$. Proofs in g will be of the form $\langle \pi, \varphi \rangle$. On such input, M_g queries all strings $\langle 1^{|\pi|}, a \rangle$, $a \in \Sigma^{\leq c \cdot \log |\pi|}$. For each negative answer, M_g simulates M_f on input π using a as advice. If any of these simulations outputs φ, then M_g also outputs φ, otherwise $g(\langle \pi, \varphi \rangle)$ is undefined. Because M_g performs at most polynomially many simulations of M_f, the machine M_g runs in polynomial time. Correctness and completeness of g follow from the fact that M_f is simulated with all correct advice strings, and the original advice used by M_f is among these (as also other advice strings are used, g might have shorter proofs than f, though).

For the second claim, let f be a propositional proof system computed by the oracle transducer M_f under the sparse NP-oracle A. Let M_A be an NP-machine for A and let $p(n)$ be a polynomial bounding the cardinality of $A \cap \Sigma^{\leq n}$ as well as the running times of M_A and M_f. With these conventions, there are at most $q(n) = p(p(n))$ many strings in A that M_f may query on inputs of length n.

We now define a machine M_g, an advice function h_g, and an advice selector ℓ_g which together yield a propositional proof system $g \geq f$ with logarithmic advice. The advice function will be $h_g(n) = |A \cap \Sigma^{\leq p(n)}|$. As A is sparse this results in logarithmic advice. Proofs in the system g are of the form

$$\pi_g = \big\langle a_1, w_1, \ldots, a_{q(n)}, w_{q(n)}, \pi_f \big\rangle$$

where $\pi_f \in \Sigma^n$ (an f-proof), $a_1, \ldots, a_{q(n)} \in \Sigma^{\leq p(n)}$ (elements from A), and $w_1, \ldots, w_{q(n)} \in \Sigma^{\leq q(n)}$ (computations of M_A). At such a proof π_g, the advice selector chooses the advice corresponding to $|\pi_f|$, i.e., we set $\ell_g(\pi_g) = |\pi_f|$. The machine M_g works as follows: it first uses its advice to obtain the number $m = h_g(|\pi_f|)$ and checks whether $a_1, \ldots, a_m$ are pairwise distinct and for each $i = 1, \ldots, m$, the string w_i is an accepting computation of M_A on input a_i. If all these simulations succeed, then we know that $A \cap \Sigma^{\leq p(n)} = \{a_1, \ldots, a_m\}$. Hence M_g can now simulate M_f on π_f and give correct answers to all oracle queries made in this computation. □

As a consequence, we get the following simplicity result stating that we can bound the complexity of the non-uniform part (the advice) when proving propositional tautologies:

Corollary 5. *Every ps*/log *f for* TAUT *is simulated by a ps*/log *g whose advice function h is computable in* $\mathsf{FP}^{\mathsf{NP}\cap\mathsf{Sparse[log]}}$*, i.e., h is computable in polynomial time with a logarithmic number of queries to a sparse* NP*-oracle.*

Proof. The claim follows by first applying item 1 and then item 2 of Theorem 4 and observing that the advice function of the resulting proof system (denoted h_g in the proof above) is computable using binary search with logarithmically many questions to the sparse NP-set $\{\, \langle 1^m, 1^n \rangle \mid m \leq |A \cap \Sigma^{\leq p(n)}| \,\}$. □

Apparently, Theorem 4 and Corollary 5 do not only hold for propositional proof systems, but for all proof systems for languages in coNP. Further, by an easy modification in the above proofs it follows that instead of a sparse NP-set it also suffices to use a tally NP-set as the oracle. Let us remark that Balcázar and Schöning [BS92] have shown a similar trade-off between advice and oracle access in complexity theory: $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{log}$ if and only if $\mathsf{coNP} \subseteq \mathsf{NP}^S$ for some sparse $S \in \mathsf{NP}$. We complete the picture by showing that the simulations in the previous theorem cannot be strengthened to a full equivalence between the two concepts:

Proposition 6. *For every language L there exist proof systems with constant advice which cannot be computed with access to a recursive oracle.*

Proof. Let us first consider the case that L is recursively enumerable and let f be a polynomial-time computable proof system for L. With each infinite sequence $a = (a_i)_{i\in\mathbb{N}}$, $a_i \in \{0,1\}$, we associate the proof system

$$f_a(\pi) = \begin{cases} f(\pi') & \text{if either } \pi = 0\pi' \text{ or } (\pi = 1\pi' \text{ and } a_{|\pi|} = 0) \\ \text{undefined} & \text{if } \pi = 1\pi' \text{ and } a_{|\pi|} = 1. \end{cases}$$

Because of the first line of its definition, f_a is a complete proof system for L. As different sequences a and b yield different proof systems f_a and f_b, there exist uncountably many different propositional proof systems with one bit of advice. On the other hand, there are only countably many proof systems computed by oracle Turing machines under recursive oracles. Hence the claim follows.

Now consider the case that L is not recursively enumerable. Yet, L has a proof system with one bit of advice which is computed by the machine M

$$M(w) = \begin{cases} x & \text{if } h(|w|) = 1 \text{ and } w = 1^x \text{ (the string } x \text{ coded in unary)} \\ \text{undef.} & \text{otherwise} \end{cases}$$

where h is the advice function for M defined as

$$h(n) = \begin{cases} 1 & \text{if } n = |1^x| \text{ and } x \in L \\ 0 & \text{otherwise.} \end{cases}$$

On the other hand, if L is not recursively enumerable, then L does not have a proof system which is computable in polynomial time under a recursive oracle. Hence the claim also holds in this case. □

For polynomial instead of logarithmic advice, we obtain a similar result as Theorem 4, but there are two differences. On the one hand, the result holds for arbitrary languages, whereas Theorem 4 only holds for languages in coNP. Also, we will now get a full equivalence between the two concepts (compare with Proposition 6). On the other hand, the oracle will still be sparse, but we cannot bound its complexity—it will be as complex as the original advice.

Proposition 7. *Let L be an arbitrary language and let f a proof system for L. Then f is a ps/poly if and only if f can be computed in polynomial time with access to a sparse oracle.*

Proof. For the first direction, let f be a proof system for L computed by the polynomial-time Turing transducer M_f with advice function h_f where $|h_f(n)| \leq p(n)$ for some polynomial p. We choose a length-injective polynomial-time computable pairing function $\langle \cdot \rangle$ and consider the set

$$A = \{ \langle 1^n, a \rangle \mid a \text{ is a prefix of } h_f(n) \} \quad .$$

Now, f can be computed in polynomial time with oracle access to A by first computing the relevant advice using prefix search and then simulating M_f.

Conversely, if f is computed in polynomial time $q(n)$ under a sparse oracle B, then f is computable by a *ps*/poly with input advice using as advice function $h(n)$ the concatenation of all strings from $B \cap \Sigma^{\leq q(n)}$. □

5 Optimal Proof Systems

A proof system for a language L which simulates every other proof system for L is called *optimal*. While in the classical setting, the existence of optimal proof systems is a prominent open question [KP89], Cook and Krajíček [CK07] have shown that there exists a propositional proof system with one bit of input advice which simulates all classical Cook-Reckhow proof systems. The proof of this result easily generalizes to arbitrary languages L, thus yielding:

Theorem 8. *For every language L there exists a proof system P with one bit of input advice such that P simulates all ps/log for L. Moreover, P p-simulates all advice-free proof systems for L.*

Proof. Let $\langle \cdot, \dots, \cdot \rangle$ be a polynomial-time computable tupling function on Σ^* which is length injective, i.e., $|\langle x_1, \dots, x_n \rangle| = |\langle y_1, \dots, y_n \rangle|$ implies $|x_i| = |y_i|$ for $i = 1, \dots, n$. We define the proof system P as follows. P-proofs are of the form $w = \langle \pi, 1^T, 1^a, 1^m \rangle$ with $\pi, T, a \in \Sigma^*$ and $m \in \mathbb{N}$ (here 1^T and 1^a denote unary encodings of T and a, respectively).

The proof system P uses one bit $h(|w|)$ of advice, where $h(|w|) = 1$ if and only if the transducer T with advice a only outputs elements from L for inputs of length $|\pi|$. Note that by the length injectivity of $\langle \cdot, \dots, \cdot \rangle$, the advice bit can in fact refer to T, a, and $|\pi|$. Now, if $h(|w|) = 1$ and T on input π with advice a outputs y after at most m steps, then $P(w) = y$. Otherwise, $P(w)$ is undefined.

In case Q is a proof system computed by some polynomial-time transducer T without (i.e. zero bits of) advice, then Q is p-simulated by P via the polynomial-time computable function $\pi \mapsto \langle \pi, 1^T, 1^\varepsilon, 1^{p(|\pi|)} \rangle$, where p is a polynomial bound for the running time of T (and ε is the empty string). On the other hand, if T uses advice $h(|\ell(\pi)|)$ of at most logarithmic length, then Q is simulated by P via the function $\pi \mapsto \langle \pi, 1^T, 1^{h(|\ell(\pi)|)}, 1^{p(|\pi|)} \rangle$. $\Box$

In contrast, it seems unlikely that we can obtain a similar result for output advice by current techniques (cf. [BM] were we investigated this problem for propositional proof systems). The question whether this optimality result can be strengthened to p-optimality (where the simulations are replaced by p-simulations) was also studied in detail in [BM], with both negative and positive results providing partial answers to the question.

Combining Theorems 4 and 8, we can reformulate the optimality result for propositional proof systems in the oracle model:

Corollary 9. *There exists a propositional proof system f which simulates every polynomial-time computable propositional proof system. The system f is computable in polynomial time under a sparse* NP*-oracle.*

Our next result shows that if advice does not help to shorten proofs even for easy languages, then optimal propositional proof systems exist.

Theorem 10. *If every polynomially bounded proof system with one bit of output advice for some $L \in \mathsf{coNP}$ can be simulated by a proof system without advice, then the class of all polynomial-time computable propositional proof systems contains an optimal system.*

Proof. Book [Boo74] showed that $\mathsf{NE} = \mathsf{E}$ if and only if any tally set $A \in \mathsf{coNP}$ belongs to NP. The former, however, implies the existence of an optimal proof system by a result of Krajíček and Pudlák [KP89]. Therefore it suffices to show that the assumption implies that any tally set $A \in \mathsf{coNP}$ belongs to NP. Since A is tally, A has a polynomially bounded propositional proof system f with one bit of output advice because we can define $f(x) = x$ if the advice $h(|x|)$ equals 1 and leave it undefined otherwise. Here, the advice function h is the characteristic function of A. Now let g be a propositional proof system without advice that simulates f. Then it follows that g is polynomially bounded and hence $A \in \mathsf{NP}$. $\Box$

Let us remark that the hypothesis in Theorem 10 does not refer to TAUT, but only to some of its subsets which are easy to prove with the help of advice.

6 Intermezzo – Nondeterministic Instance Complexity

Before we can continue our investigation of proof systems with advice and approach question Q1 on the existence of polynomially bounded proof systems, we need to review the notion of nondeterministic instance complexity and prove some new facts on this complexity measure.

While Kolmogorov complexity studies the hardness of individual strings, the notion of instance complexity was introduced by Orponen, Ko, Schöning, and Watanabe [OKSW94] to measure the hardness of individual instances of a given language. The deterministic instance complexity of [OKSW94] was later generalized to the nondeterministic setting by Arvind, Köbler, Mundhenk, and Torán [AKMT00].

As required for Kolmogorov complexity and instance complexity, we fix a universal Turing machine $U(M, x)$ which executes nondeterministic programs M on inputs x. In the sequel, we refrain from always mentioning U explicitly. Thus we simply write statements like "M is a t-time bounded Turing machine" with the precise meaning that U always spends at most $t(n)$ steps to simulate M on inputs of length n. Likewise, to "simulate a machine M on input x" always means executing $U(M, x)$.

A nondeterministic Turing machine M is *consistent* with a language L (or L-consistent), if $L(M) \subseteq L$. We can now give the definition of nondeterministic instance complexity from [AKMT00].

Definition 11 (Arvind et al. [AKMT00]). *For a set L and a time bound t, the t-time-bounded* nondeterministic instance complexity *of x with respect to L is defined as*

$$nic^t(x : L) = \min\{\, |M| \ : M \text{ is an } L\text{-consistent } t\text{-time-bounded nondeterministic machine, and } M \text{ decides correctly on } x \,\} \ .$$

Similarly as in the deterministic case in [OKSW94], we collect all languages with prescribed upper bounds on the running time and nondeterministic instance complexity in a complexity class.

Definition 12. *Let F_1 and F_2 be two classes of functions. We define*

$$\mathsf{NIC}[F_1, F_2] = \{L : \text{there exist } s \in F_1 \text{ and } t \in F_2 \text{ such that for all } x \in \Sigma^* \ nic^t(x : L) \leq s(|x|)\} \ .$$

A particularly interesting choice for the classes F_1 and F_2 is to allow polynomial running time, but only logarithmic descriptions for the machines. This leads to the class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ which plays a central role in this paper. Similarly as in the deterministic case (cf. [OKSW94]), the next proposition locates this class between the nonuniform classes $\mathsf{NP/log}$ and $\mathsf{NP/poly}$.

Proposition 13. $\mathsf{NP/log} \subseteq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subseteq \mathsf{NP/poly}$.

Proof. For the first inclusion, let $L \in \mathsf{NP/log}$. Let M be a nondeterministic Turing machine with logarithmic advice that decides L and let a_n be the advice given to M for inputs of length n. We define a collection of programs M_{n,a_n} for L as follows. On input x the machine M_{n,a_n} first checks, whether the length of the input is n. For this we need to code the number n into M_{n,a_n}. If $|x| \neq n$, then M_{n,a_n} rejects. Otherwise, M_{n,a_n} simulates M on input x with advice a_n which is also coded into M_{n,a_n}. Essentially, the machines M_{n,a_n} are constructed by hardwiring n and a_n into M, and thus the size of M_{n,a_n} is logarithmic in n. Therefore $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the second inclusion, let $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Then there exist a constant c and a polynomial p such that for all x we have $nic^p(x : L) \leq c \log |x| + c$. We construct a nondeterministic Turing machine M with polynomial advice that accepts exactly L. The advice of M for length n consists of all nondeterministic Turing machines $M_1, \ldots, M_m$ of size at most $c \log n + c$ which are consistent with L. Note that for each input length n, there are only polynomially many machines of the appropriate size $\leq c \log n + c$. Hence polynomial advice suffices to encode the whole list $M_1, \ldots, M_m$. On input x, the machine M simulates each M_i on x for at most $p(|x|)$ steps. If any of the M_i accepts, then M accepts as well, otherwise it rejects.

We claim, that $L(M) = L$. For, if $x \in L$, then there is a nondeterministic L-consistent Turing machine M_i such that $M_i(x)$ accepts and $|M_i| \leq c \log |x| + c$. Thus, also $M(x)$ accepts. If, on the other hand, M accepts x, then so does some M_i which is consistent with L. Therefore, $x \in L$ because $L(M_i) \subseteq L$. $\square$

In fact, the inclusions in Proposition 13 are proper as we will show in Theorem 15 below. For the proof we need the following notion:

Definition 14 (Buhrman, Fortnow, Laplante [BFL01]). *For a time bound t, the* nondeterministic decision complexity of x, *denoted $CND^t(x)$, is the minimal size of a t-time-bounded nondeterministic Turing machine M with $L(M) = \{x\}$.*

As already noted in [AKMT00], the *CND* measure provides an upper bound to the *nic* measure, i.e., for any language L and time bound t there is a constant $c > 0$ such that $nic^t(x : L) \leq CND^t(x) + c$ for all $x \in \Sigma^*$. By a simple counting argument, it follows that for any length n there exist strings x of length n with $CND(x) \geq n$, where $CND(x)$ is the minimal size of a nondeterministic Turing machine M with $L(M) = \{x\}$ (i.e., the time-unbounded *CND* measure).

Inspired by a similar result in [OKSW94], we now prove the following separations:

Theorem 15. *1. For every constant $c > 0$, $\mathsf{NP}/n^c \not\supseteq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.*
2. $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \not\supseteq \mathsf{P}/\mathsf{lin}$.

Proof. For the first item, let $0 < c < d$ be natural numbers. Diagonalizing against all NP machines and all advice strings, we inductively define a set A with $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, but $A \notin \mathsf{NP}/n^c$. Let $(N_i)_{i \in \mathbb{N}}$ be an enumeration of all NP machines, in which every machine occurs infinitely often. In step n we diagonalize against the machine N_n and every advice string of length $\leq n^c$ which N_n might use for length n. Let $x_1, \ldots, x_{2^n}$ be the lexicographic enumeration of all strings in Σ^n and let $S_n = \{x_1, \ldots, x_{n^d}\} \subseteq \Sigma^n$. For each string w of length at most n^c, let $A_w = \{x \in S_n : N_n(x) \text{ accepts under advice } w\}$. Since there are only 2^{n^c} such sets, but 2^{n^d} subsets of S_n, there must be one which is not equal to any A_w. For every n, let A_n be one such set, and let $A = \bigcup_n A_n$. By construction, $A \notin \mathsf{NP}/n^c$.

We still have to show $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. For each string s, let $\widetilde{s}$ be the substring of s which has all leading zeros deleted. For each n and each $a \in A_n$, let $M_{n,\widetilde{a}}$ be the following machine: on input x, the machine $M_{n,\widetilde{a}}$ checks whether $|x| = n$ and $\widetilde{x} = \widetilde{a}$. If this test is positive, then $M_{n,\widetilde{a}}$ accepts, otherwise it rejects. The machine $M_{n,\widetilde{a}}$ is of size $O(\log n)$, as both n and $\widetilde{a}$ are of length $O(\log n)$

(Observe that the first n^d elements in the lexicographic order of Σ^n have no 1's appearing before the last $\log n^d$ bits). Thus $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the second item, let A be a set that contains exactly one element x per length with $\mathit{CND}(x) \geq |x|$. Obviously, $A \in \mathsf{P/lin}$ because A contains exactly one string per length and this element can be given as advice. On the other hand, $A \notin \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Assume on the contrary, that $A \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Then there are a constant c and a polynomial p, such that for each $x \in A$, there is an A-consistent p-time-bounded machine M_x of size $\leq c \log |x| + c$ which accepts x. We modify M_x to a machine M'_x such that $L(M'_x) = \{x\}$ and $|M'_x| \leq c' \log |x| + c'$ for some constant c'. This machine M'_x works as follows: on input y, the machine M'_x first checks, whether $|y| = |x|$. If not, it rejects. Otherwise, it simulates $M_x(y)$. Thus for all $x \in A$, $\mathit{CND}(x) \leq c' \log |x| + c'$, contradicting the choice of A. □

From Theorem 15 we infer that both inclusions in Proposition 13 are strict:

Corollary 16. $\mathsf{NP/log} \subsetneq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subsetneq \mathsf{NP/poly}$.

7 Polynomially Bounded Proof Systems with Advice

For any language L, we now investigate question Q1 whether L has a polynomially bounded proof system with advice. We obtain different characterizations of this question, depending on

- whether we use input or output advice,
- which amount of advice the proof system may use, and
- the complexity of the proven language L.

We first consider proof systems with output advice. Similarly as in the classical result by Cook and Reckhow [CR79], we obtain the following equivalence:

Theorem 17. *Let $L \subseteq \Sigma^*$ be a language and let $k : \mathbb{N} \to \mathbb{N}$ be a function. Then L has a polynomially bounded ps/k with output advice if and only if $L \in \mathsf{NP}/k$.*

Proof. For the forward implication, let P be a polynomially bounded ps/k with output advice for L and let p be a bounding polynomial for P. We construct an NP/k machine M which uses the same advice as P and decides L. On input x, the machine M guesses a P proof w of size $\leq p(|x|)$ and checks whether $P(w) = x$. If so, M accepts, otherwise M rejects.

For the backward implication, let N be an NP/k machine deciding L with advice function h. We define a proof system P for L with k bits of output advice. Again, both P and N use the same advice. On input $\pi = \langle w, x \rangle$ the proof system P checks, whether w is an accepting computation of N on input x with advice $h(|x|)$. If so, then $P(\pi) = x$. Otherwise, $P(\pi)$ is undefined. □

Given this result, we can now concentrate on input advice. In view of Theorem 8, input advice appears to be a stronger concept than output advice (as we probably cannot expect a similar result as Theorem 8 for output advice, cf. [BM] and also Corollary 21 and Proposition 25 below for further results supporting this claim). Surprisingly, the advantage of input advice seems to vanish when we allow a polynomial amount of advice.

Theorem 18. *Let $L \subseteq \Sigma^*$ be any language. Then L has a polynomially bounded ps/poly with output advice if and only if L has a polynomially bounded ps/poly with input advice.*

Proof. The forward direction is a simple application of Proposition 3.

For the backward implication, let f_{in} be a ps/poly with input advice for L bounded by some polynomial p. Let a_n be the polynomially length-bounded advice used by f_{in} on inputs of length n.

We define a polynomially bounded ps/poly f_{out} for L with output advice as follows. Inputs x for f_{out} are interpreted as pairs $x = \langle \pi, y \rangle$. If $|\pi| \leq p(|y|)$ and $f_{in}(\pi) = y$, then $f_{out}(x) = y$. Otherwise, f_{out} is undefined. The computation of f_{out} uses all advice strings for f_{in} up to length $p(|y|)$ as advice. This still results in polynomial-size output advice for f_{out}.

The system f_{out} is correct, because f_{in} is correct. It is complete, because every $y \in L$ has a proof π_y with $|\pi_y| \leq p(|y|)$, implying that $f_{out}(\langle \pi_y, y \rangle) = y$. Hence, f_{out} is a polynomially bounded ps/poly with output advice. □

By Theorems 17 and 18, the existence of polynomially bounded ps/poly with input advice for L is equivalent to $L \in \mathsf{NP/poly}$. Next, we consider proof systems with only a logarithmic amount of advice. In this case, we get a similar equivalence as before, where the class $\mathsf{NP/poly}$ is replaced by the instance complexity class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

Theorem 19. *For every language L the following conditions are equivalent:*

1. *L has a polynomially bounded $ps/1$ with input advice.*
2. *L has a polynomially bounded ps/log with input advice.*
3. *$L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.*

Proof. The implication *1 ⇒ 2* follows by definition.

To prove the implication *2 ⇒ 3*, let f be a polynomially bounded ps/log with input advice and bounding polynomial p. For each x we have to construct a program M which is consistent with L and correctly decides x. If $x \notin L$, then M can just always reject. If $x \in L$, then there exists an f-proof π of x of length $\leq p(|x|)$. Let a be the advice for f on inputs of length $|\pi|$. To construct the machine M for x, we hardwire the values of $|x|$, $|\pi|$, and a into M. On input y the machine M checks, whether $|y| = |x|$. If not, it rejects. Otherwise M guesses an f-proof π' of length $|\pi|$ for y and verifies that $f(\pi') = y$ using the advice a. If this test is positive, then M accepts, otherwise M rejects. Clearly, M accepts exactly all elements from L of length $|x|$ which have f-proofs of length $|\pi|$. In particular, M accepts x. Additionally, M is a polynomial-time nondeterministic program of length at most $c + \log|x| + \log|\pi| + |a|$ for some constant c. Therefore $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

For the remaining implication *3 ⇒ 1*, let us assume that there are a polynomial p and a constant c, such that for every x, $nic^p(x : L) \leq c \cdot \log(|x|) + c$. We define a polynomially bounded $ps/1$ f for L with input advice as follows. Proofs in f take the form $\pi = \langle x, w, 1^M \rangle$, where $\langle \cdot, \dots, \cdot \rangle$ is a polynomial-time computable and length-injective tupling function. The advice for f certifies whether or not M is a polynomial-time Turing machine that is consistent with

L. If this is the case and w is an accepting computation of M on input x, then $f(\pi) = x$. Otherwise, $f(\pi)$ is undefined. Note that in the proof π we described the machine M in tally form. Together with the length-injectivity of the tupling function this allows the advice to refer to the machine M (but not to the input x which is given in binary notation).

Now, since $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, for every $x \in L$ there is an L-consistent Turing machine M_x with running time p which accepts x and $|M_x| \leq c \cdot \log |x| + c$. Thus every element $x \in L$ has a polynomial-size f-proof $\langle x, w, 1^{M_x} \rangle$ where w is an accepting path of $M_x(x)$. $\square$

In fact, we can prove a more general version of the preceding theorem, where we replace polynomial upper bounds for the proof length by arbitrary upper bounds. In this way we obtain:

Theorem 20. *For any language L and any function $t : \mathbb{N} \to \mathbb{N}$, $t \in n^{\Omega(1)}$, the following conditions are equivalent:*

1. *L has a $t^{O(1)}$-bounded ps/1 with input advice.*
2. *L has a $t^{O(1)}$-bounded ps/log with input advice.*
3. *$L \in \mathsf{NIC}[O(\log t), t^{O(1)}]$.*

For a language L we now consider the following three assertions:

A1: L has a polynomially bounded *ps*/log with output advice.
A2: L has a polynomially bounded *ps*/log with input advice.
A3: L has a polynomially bounded *ps*/poly with output advice.

By our results so far, assertions A1, A2, and A3 are equivalent to the statement that L is contained in the classes NP/log, $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, and $\mathsf{NP}/\mathsf{poly}$, respectively. As these classes form a chain of inclusions by Proposition 13, we get the implications A1 $\Rightarrow$ A2 $\Rightarrow$ A3 for every L. Moreover, by Corollary 16, the inclusions $\mathsf{NP}/\mathsf{log} \subsetneq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subsetneq \mathsf{NP}/\mathsf{poly}$ are proper. Hence we obtain:

Corollary 21. *There exist languages L for which* A2 *is fulfilled, but* A1 *fails. Likewise, there exist languages L for which* A3 *is fulfilled, but* A2 *fails.*

Table 1 provides an overview of our results on question Q1 obtained so far, showing which languages possess polynomially bounded proof systems with advice. It is interesting to note that all languages appearing in this table form a chain of strict inclusions (cf. Corollary 16).

8 Polynomially Bounded Proof Systems for TAUT

From a practical point of view, it is most interesting to investigate what precisely happens for $L = \mathrm{TAUT}$ (or more generally for problems in coNP). Even though by Corollary 16, NP/log and $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ are distinct, they do not differ inside coNP, as the next theorem shows.

Theorem 22. *Let $L \in \mathsf{coNP}$. Then $L \in \mathsf{NP}/\mathsf{log}$ if and only if $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Moreover, if $L \in \mathsf{NP}/\mathsf{log}$, then the advice can be computed in $\mathsf{FP}^{\mathsf{NP}[\mathsf{log}]}$.*

Table 1. Languages with polynomially bounded proof systems

	input advice	output advice
ps/poly	$\mathsf{NP}/\mathsf{poly}$	$\mathsf{NP}/\mathsf{poly}$
ps/log	$\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$	NP/log
$ps/1$	$\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$	$\mathsf{NP}/1$
$ps/0$	NP	

Proof. By Proposition 13 we only have to prove the backward implication. For this let L be a language from coNP. Assuming $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$, there exists a polynomial p and a constant c such that $nic^p(x : L) \leq c \log |x| + c$ for all $x \in \Sigma^*$. Let Π^n be the set of all p-time bounded nondeterministic machines M with $|M| \leq c \log n + c$. Let further a_n be the number of machines from Π^n that are not consistent with $L \cap \Sigma^{\leq n}$. As the cardinality of Π^n is bounded by a polynomial in n, the length of the number a_n is logarithmic in n.

We now construct a nondeterministic Turing machine N that uses $c \log n + c + 1$ bits of advice for inputs of length n and decides L. The advice of N for input length n will be the number a_n. On input x of length n, the machine N nondeterministically chooses a_n pairwise distinct machines $M_1, \dots, M_{a_n} \in \Pi^n$ and strings $x_1, \dots, x_{a_n} \in \Sigma^{\leq n}$. Next, N verifies that $x_1, \dots, x_{a_n}$ do not belong to L. As $L \in \mathsf{coNP}$, this can be done in nondeterministic polynomial time. Then N checks whether for each $i = 1, \dots, a_n$ the machine M_i accepts the input x_i. If any of the tests so far failed, N rejects. Otherwise, if all these tests were positive, we know that every machine in $\Pi^n \setminus \{M_1, \dots, M_{a_n}\}$ is consistent with $L \cap \Sigma^{\leq n}$. After this verification has successfully taken place, N simulates all remaining machines $M \in \Pi^n \setminus \{M_1, \dots, M_{a_n}\}$ on input x. If one of these simulations accepts, then also N accepts x, otherwise N rejects.

Since there are only consistent machines left after a_n machines have been deleted, N never accepts any $x \notin L$. On the other hand, the assumption $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ guarantees that for every $x \in L$ there is a machine in Π^n which is consistent with L and accepts x. Therefore N correctly decides L, and thus $L \in \mathsf{NP}/\mathsf{log}$, as claimed.

For the additional claim in the theorem, it suffices to observe that using binary search we can compute the advice a_n with at most logarithmically many queries of the form "Do there exist at least m logarithmic-size machines which are inconsistent with $L \cap \Sigma^{\leq n}$?" As this is an NP question, the advice can be computed in $\mathsf{FP}^{\mathsf{NP}[\mathsf{log}]}$. □

By Theorem 18 we already know that TAUT has a polynomially bounded ps/poly with input advice if and only if it has a polynomially bounded ps/poly with output advice. As a corollary to Theorem 22 we obtain the same equivalence for logarithmic advice.

Corollary 23. TAUT *has a polynomially bounded* ps/log *with input advice if and only if* TAUT *has a polynomially bounded* ps/log *with output advice.*

Descending to constant advice, this equivalence seems to fail, as we show below. For this we use a result of Buhrman, Chang, and Fortnow [BCF03]:

Theorem 24 (Buhrman, Chang, Fortnow [BCF03]). *For every constant* $k \geq 1$, $\mathsf{coNP} \subseteq \mathsf{NP}/k$ *if and only if* $\mathsf{PH} \subseteq \mathsf{BH}_{2^k}$.

Using this result we conclude that the assertions of the existence of polynomially bounded proof systems with input and output advice appear to be of different strength, as otherwise the equivalence of two collapses of PH of presumably different strength follows.

Proposition 25. *Assume that* TAUT *having a polynomially bounded* $ps/1$ *with input advice implies that* TAUT *has a polynomially bounded* $ps/1$ *with output advice. Then* $\mathsf{PH} \subseteq \mathsf{BH}$ *already implies* $\mathsf{PH} \subseteq \mathsf{D^p}$.

Proof. If the polynomial hierarchy collapses to the Boolean hierarchy, then PH in fact collapses to some level BH_k of BH. By Theorem 24, this means that $\mathsf{coNP} \subseteq \mathsf{NP}/k'$ for some constant k'. Hence by Theorem 17, TAUT has a polynomially bounded ps/k' P with output advice. By Theorem 8, this proof system P is simulated by a proof system P' which only uses 1 bit of input advice. As P is polynomially bounded, this is also true for P'. By our assumption, TAUT also has polynomially bounded $ps/1$ with output advice. By Theorem 17 this implies $\mathsf{coNP} \subseteq \mathsf{NP}/1$ and therefore $\mathsf{PH} \subseteq \mathsf{D^p}$ by Theorem 24. □

So far we have provided different characterizations of question Q1 whether polynomially bounded proof systems with advice exist. At this point it is natural to ask, how likely these assumptions actually are, i.e., what consequences follow from the assumption that such proof systems exist. For TAUT we obtain a series of collapse consequences of presumably different strength as shown in Table 2.

Table 2. Consequences of the existence of polynomially bounded proof systems

Assumption *if* TAUT *has a polynomially bounded …*		**Consequence** *then* PH *collapses to …*
ps/poly	(input or output advice)	$\mathsf{S_2^{NP}} \subseteq \mathsf{\Sigma_3^p}$
ps/log	(input or output advice)	$\mathsf{P^{NP[log]}}$
$ps/O(1)$	(input advice)	$\mathsf{P^{NP[log]}}$
$ps/O(1)$	(output advice)	$\mathsf{P^{NP[O(1)]}} = \mathsf{BH}$
$ps/0$	(no advice)	NP

The first line in Table 2 follows from Theorems 17 and 18 and a result of Cai, Chakaravarthy, Hemaspaandra, and Ogihara [CCHO05], who have shown that $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ implies $\mathsf{PH} \subseteq \mathsf{S_2^{NP}}$. For the second line, the distinction between input and output advice is again irrelevant (Corollary 23). Here we use a result of Arvind, Köbler, Mundhenk, and Torán [AKMT00], who showed that TAUT $\in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ implies $\mathsf{PH} \subseteq \mathsf{P^{NP[log]}}$. Finally, the constant-advice case (lines 3 and 4), follows from Theorem 24 in conjunction with Theorems 17 and 19. In comparison, the classical Cook-Reckhow Theorem states that TAUT has an advice-free polynomially bounded proof system if and only if $\mathsf{PH} \subseteq \mathsf{NP}$ (line 5).

9 Simplifying the Advice in Propositional Proof Systems

In this final section we again concentrate on propositional proof systems and prove a result which contributes to an answer to our last question Q3. There are two natural ways to enhance proof systems with advice by either supplying non-uniform information to the proof (input advice) or to the proven formula (output advice). Intuitively, input advice is the stronger model: proofs can be quite long and formulas of the same size typically require proofs of different size. Hence, supplying advice depending on the proof size is not only more flexible, but also results in more advice per formula. This view is also supported by our results obtained so far: there exist optimal proof systems with input advice (Theorem 8), whereas for output advice a similar result cannot be obtained by current techniques [BM]. Further evidence is provided by the existence of languages that have polynomially bounded proof systems with logarithmic input advice, but do not have such systems with output advice (Corollary 21).

In our next result we show how input advice can be transformed into output advice. We obtain this simplification of advice under the assumption of weak, but non-trivial upper bounds to the proof size. More precisely, from a propositional proof system which uses logarithmic input advice and has sub-exponential size proofs of all tautologies, we construct a system with polynomial output advice which obeys almost the same upper bounds. For the proof we use a new technique by Buhrman and Hitchcock [BH08] who show that sets of sub-exponential density are not NP-hard unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

Theorem 26. *Let $t(n) \in 2^{O(\sqrt{n})}$ and assume that there exists a $t(n)$-bounded propositional proof system f with polylogarithmic input advice. Then there exists an $s(n)$-bounded propositional proof system g with polynomial output advice where $s(n) \in O(t(d \cdot n^2))$ with some fixed constant d independent of f.*

Proof. Let $t(n) \le 2^{c\cdot\sqrt{n}}$ for some constant c and let f be a $t(n)$-bounded propositional proof system with polylogarithmic input advice. We say that π is a *conjunctive f-proof* for a tautology φ if there exist tautologies $\psi_1, \dots, \psi_n$ with $|\psi_i| = |\varphi| = n$ such that $f(\pi) = \psi_1 \wedge \dots \wedge \psi_n$ and φ is among the ψ_i. For a number $m \ge 1$, we denote by $\sharp^n_m$ the number of tautologies $\varphi \in \mathrm{TAUT}^{=n}$ which have conjunctive f-proofs of size exactly m. By counting we obtain

$$(\sharp^n_m)^n \ge |\{(\varphi_1, \dots, \varphi_n) \mid \varphi_1 \wedge \dots \wedge \varphi_n \text{ has an } f\text{-proof of size } m \text{ and } |\varphi_i| = n \text{ for } 1 \le i \le n\}| \ . \tag{1}$$

As f is t-bounded, every $\varphi \in \mathrm{TAUT}^{=n}$ has a conjunctive f-proof of size at most $t(d \cdot n^2)$ where d is a constant such that for each sequence $\psi_1, \dots, \psi_n$ of formulas of length n, $|\psi_1 \wedge \dots \wedge \psi_n| \le d \cdot n^2$. Let $\sharp^n = \max\{\sharp^n_m \mid m \le t(d \cdot n^2)\}$.

Using (1) we obtain

$$
\begin{aligned}
|\mathrm{TAUT}^{=n}|^n \leq \sum_{m=1}^{t(d\cdot n^2)} (\sharp^n_m)^n \;&\leq\; (\sharp^n)^n \cdot t(d\cdot n^2) \\
&\leq (\sharp^n)^n \cdot 2^{c\cdot\sqrt{d\cdot n^2}} \;=\; (\sharp^n \cdot 2^{c\cdot\sqrt{d}})^n \ .
\end{aligned}
$$

Thus, setting $\delta = 2^{-c\cdot\sqrt{d}}$, we get $\sharp^n \geq \delta\cdot|\mathrm{TAUT}^{=n}|$. Therefore, by definition of $\sharp^n$ there exists a number $m_{n,0} \leq t(d\cdot n^2)$ such that $\sharp^n_{m_{n,0}} \geq \delta\cdot|\mathrm{TAUT}^{=n}|$, i.e., a δ-fraction of all tautologies of length n has a conjunctive f-proof of size $m_{n,0}$.

Consider now the set $\mathrm{TAUT}_0^{=n}$ of all tautologies of length n which do not have conjunctive f-proofs of size $m_{n,0}$. Repeating the above argument for $\mathrm{TAUT}_0^{=n}$ yields a proof length $m_{n,1}$ such that $\sharp^n_{m_{n,1}} \geq \delta\cdot|\mathrm{TAUT}_0^{=n}|$. Iterating this argument we obtain a sequence

$$
m_{n,0}, m_{n,1}, \ldots, m_{n,\ell(n)} \qquad \text{with } \ell(n) = \left\lceil \frac{\log|\mathrm{TAUT}^{=n}|}{\log(1-\delta)^{-1}} \right\rceil \leq \left\lceil \frac{n}{\log(1-\delta)^{-1}} \right\rceil
$$

such that every $\varphi \in \mathrm{TAUT}^{=n}$ has a conjunctive f proof of size $m_{n,i}$ for some $i \in \{0,\ldots,\ell(n)\}$.

We will now define a proof system g which uses polynomial output advice and obeys the same proof lengths as f. Assume that f is computed by the polynomial-time Turing transducer M_f with advice function h_f. The system g will be computed by a polynomial-time Turing transducer M_g using the advice function

$$
h_g(n) = \big\langle m_{n,0}, h_f(m_{n,0}), \ldots, m_{n,\ell(n)}, h_f(m_{n,\ell(n)})\big\rangle \ .
$$

The machine M_g works as follows: first M_g checks whether the proof has the form

$$
\langle \varphi, \psi_1, \ldots, \psi_n, \pi, i\rangle
$$

where $\varphi, \psi_1, \ldots, \psi_n$ are formulas of length n such that $\varphi \in \{\psi_1,\ldots,\psi_n\}$, π is a string (an f-proof), and i is a number $\leq \ell(n)$. If this test fails, then M_g outputs $\top^n$ (an easy tautology of length n). Then M_g uses its advice to check whether $|\pi| = m_{n,i}$. If so, then M_g simulates M_f on input π using advice $h_f(m_{n,i})$ (which is available through the advice function h_g). If the output of this simulation is $\psi_1 \wedge \cdots \wedge \psi_n$, then M_g outputs φ, otherwise it outputs $\top^n$.

By our analysis above, g is a propositional proof system (it is correct and complete). The advice only depends on the length n of the proven formula, so g uses output advice. To estimate the advice length, let $|h_f(m)| \leq \log^a m$ for some constant a. Then

$$
|h_g(n)| \leq \sum_{i=0}^{\ell(n)} (|m_{n,i}| + |h(m_{n,i}|) \leq (\ell(n)+1)\left(c\sqrt{n} + \log^a(2^{c\sqrt{n}})\right) \in n^{O(1)} \ .
$$

The size of a g-proof $\langle\varphi,\psi_1,\ldots,\psi_n,\pi,i\rangle$ for $\varphi \in \mathrm{TAUT}^{=n}$ is dominated by $|\pi| \leq t(d\cdot n^2)$, and therefore g is $s(n)$-bounded for some $s(n) \in O(t(d\cdot n^2))$. $\square$

In some sense, Theorem 26 transfers the results of Theorem 18 and Corollary 23 to super-polynomial proof lengths. However, while Theorem 18 has an easy proof and holds for all languages, the last construction is rather non-trivial and uses some assumption on L. Here we stated the result for the most interesting case $L = \text{TAUT}$, but the same proof also works for all languages L with a polynomial-time computable AND-function.

10 Conclusion

In this paper we have addressed some fundamental questions on proof systems in the new advice model. From a practical perspective, propositional proof systems with advice form the most interesting instances. Undoubtedly, the main question is: Does advice help to prove propositional tautologies? In this generality, we leave open the question—but our results provide partial answers. On the one hand, when proving tautologies "very complicated" advice is not necessary—it suffices to use a "small amount of simple" advice (Theorem 4). Further, if advice is helpful to prove tautologies in the sense that proofs become shorter in general, then again the advice can be simplified (Theorem 26).

On the other hand, if advice is not at all useful to prove tautologies, then optimal propositional proof systems exist (Theorem 10), a consequence which is considered unlikely by many researchers (cf. [KMT03]). For further research, it seems interesting to explore how natural proof systems like resolution can facilitate advice. Is it possible to shorten proofs in such systems by using advice?

Acknowledgement. The first author wishes to thank Uwe Schöning for suggesting to apply results from [BS92] in the context of proof systems with advice. We also thank the anonymous referees of the conference versions for helpful comments and detailed suggestions on how to improve this paper.

References

[AKMT00] V. Arvind, Johannes Köbler, Martin Mundhenk, and Jacobo Torán. Nondeterministic instance complexity and hard-to-prove tautologies. In *Proc. 17th Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 314–323. Springer-Verlag, Berlin Heidelberg, 2000.

[BCF03] Harry Buhrman, Richard Chang, and Lance Fortnow. One bit of advice. In *Proc. 20th Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 547–558. Springer-Verlag, Berlin Heidelberg, 2003.

[BDG88] José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*. Springer-Verlag, Berlin Heidelberg, 1988.

[BFL01] Harry Buhrman, Lance Fortnow, and Sophie Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2001.

[BH08] Harry Buhrman and John M. Hitchcock. NP-hard sets are exponentially dense unless coNP $\subseteq$ NP/poly. In *Proc. 23rd Annual IEEE Conference on Computational Complexity*, pages 1–7, 2008.

[BKM09] Olaf Beyersdorff, Johannes Köbler, and Sebastian Müller. Nondeterministic instance complexity and proof systems with advice. In *Proc. 3rd International Conference on Language and Automata Theory and Applications*, volume 5457 of *Lecture Notes in Computer Science*, pages 164 – 175. Springer-Verlag, Berlin Heidelberg, 2009.

[BM] Olaf Beyersdorff and Sebastian Müller. A tight Karp-Lipton collapse result in bounded arithmetic. *ACM Transactions on Computational Logic*. To appear.

[BM09] Olaf Beyersdorff and Sebastian Müller. Does advice help to prove propositional tautologies? In *Proc. 12th International Conference on Theory and Applications of Satisfiability Testing*, volume 5584 of *Lecture Notes in Computer Science*, pages 65 – 72. Springer-Verlag, Berlin Heidelberg, 2009.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[BS92] J.L. Balcázar and U. Schöning. Logarithmic advice classes. *Theoretical Computer Science*, 99:279–290, 1992.

[CCHO05] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.

[CK07] Stephen A. Cook and Jan Krajíček. Consequences of the provability of NP $\subseteq$ P/poly. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 302–309. ACM Press, 1980.

[KMT03] Johannes Köbler, Jochen Messner, and Jacobo Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.

[KP89] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[OKSW94] P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the ACM*, 41(1):96–121, 1994.

[Pud98] Pavel Pudlák. The lengths of proofs. In Samuel R. Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, Amsterdam, 1998.