Electronic Colloquium on Computational Complexity, Report No. 75 (2008)

Nondeterministic Instance Complexity and Proof Systems with 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. Motivated by strong Karp-Lipton collapse results in bounded arithmetic, Cook and Krajíček [7] have recently introduced the notion of propositional proof systems with advice. In this paper we investigate the following question: *Do there exist polynomially bounded proof systems with advice for arbitrary languages?* Depending on the complexity of the underlying language 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 for a language L, the above question is tightly linked with the question whether L has small nondeterministic instance complexity.

1 Introduction

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. [13]).

Motivated by strong Karp-Lipton collapse results in bounded arithmetic, Cook and Krajíček [7] have recently introduced the notion of propositional proof systems using advice. This model seems to be strictly more powerful than classical proof systems, as long-standing open problems, such as the existence of optimal proof systems, receive affirmative answers in this setting [7, 3].

In the present paper we focus on the question whether there exist polynomially bounded proof systems with advice. We do not only consider propositional proof systems, but investigate this question for arbitrary proof systems and languages. As in the Cook-Reckhow Theorem above, we obtain a series of results which provide a complete complexity-theoretic characterization for this question.

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 [12]. In its nondeterministic version, Arvind, Köbler, Mundhenk,

⋆ Supported by DFG grant KO 1053/5-2

ISSN 1433-8092

and Torán [1] 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.*

To achieve these results, we start in Section 3 by reviewing the notion of nondeterministic instance complexity of [1]. Nondeterministic instance complexity is measured by two parameters: the size of the machine and its running time. The most interesting choice for these parameters seems to allow logarithmic size programs with polynomial running time. We combine all languages which admit such programs in the class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Using a proof idea from [12], we locate $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ between the advice classes NP/log and $\mathsf{NP}/\mathsf{poly}$, where these inclusions are shown to be strict.

In Section 4, we generalize the notion of propositional proof systems with advice of Cook and Krajíček [7] to arbitrary languages. We classify these proof systems on whether the advice depends on the proof (input advice) or on the proven element (output advice). Similarly as in [7], we show that for every language L, the class of all proof systems for L using logarithmic input advice contains an optimal proof system.

Our main results follow in Sections 5 and 6 were we examine the question whether polynomially bounded proof systems with advice exist. In Section 5 we investigate this problem for arbitrary languages, whereas in Section 6 we focus on TAUT which presents the most interesting case for practical applications.

For output advice, the classical Cook-Reckhow Theorem generalizes in a straightforward manner, and thus a language L has a polynomially bounded proof system with $k(n)$ bits of output advice if and only if $L \in \mathsf{NP}/k(n)$.

For input advice, which yields a strictly more powerful model, this question is more intricate. Using a polynomial amount of advice, the difference vanishes: if a language L has a polynomially bounded proof system with polynomial input advice, then L already has such a system with output advice.

Descending to logarithmic advice, we establish the connection to nondeterministic instance complexity: a language L has a polynomially bounded proof system with logarithmic input advice if and only if $L \in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$. Because $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ and NP/log are different classes, there exist languages that have a polynomially bounded proof system with logarithmic input advice, but do not have a system of this kind with output advice. Surprisingly, for TAUT the equivalence between input and output advice is again valid at the logarithmic level. Moreover, if TAUT actually has such a polynomially bounded system, i.e., if TAUT $\in \mathsf{NP}/\mathsf{log}$, then the advice is computable in $\mathsf{FP}^{\mathsf{NP}[\mathsf{log}]}$.

Reducing the amount of advice even further to constantly many bits, the equivalence between input and output advice seems to fail for TAUT, as it implies unlikely collapse consequences. Finally, we show that the actual existence of polynomially bounded advice proof systems for TAUT produces different collapses of the polynomial hierarchy.

2 Preliminaries

We assume familiarity with standard complexity classes (cf. [2]). 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 $\mathsf{P^{NP[}}O(1)\mathsf{]}$, 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 $\mathsf{P^{NP[log]}}$.

Complexity classes with *advice* were first considered by Karp and Lipton [9]. 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 $\mathsf{C}/F = \{L/h \mid L \in \mathsf{C}, h \in F\}$.

Cook and Reckhow [8] defined the notion of a *proof system* for an arbitrary language L quite generally as a partial polynomial-time computable function f with range L. A string w with $f(w) = x$ is called an f-proof for $x \in L$.

Proof systems are compared according to their strength by simulations as introduced in [8] and [10]. 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*. A proof system for L is called *(p-)optimal* if it (p-)simulates all proof systems for L. For a function $t : \mathbb{N} \to \mathbb{N}$, a proof system f is *t-bounded* if for all $x \in L$ there exists an f-proof of size at most $t(|x|)$. If t is a polynomial, then f is called *polynomially bounded*.

3 Nondeterministic Instance Complexity

While Kolmogorov complexity studies the hardness of individual strings (cf. [11]), the notion of instance complexity was introduced by Orponen, Ko, Schöning, and Watanabe [12] to measure the hardness of individual instances of a given language. The deterministic instance complexity of [12] was later generalized to the nondeterministic setting by Arvind, Köbler, Mundhenk, and Torán [1].

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 [1].

Definition 1 (Arvind et al. [1]). *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 [12], we collect all languages with prescribed upper bounds on the running time and nondeterministic instance complexity in a complexity class.

Definition 2. *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. [12]), the next proposition locates this class between the advice classes NP/log and $\mathsf{NP}/\mathsf{poly}$.

Proposition 3. $\mathsf{NP}/\mathsf{log} \subseteq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subseteq \mathsf{NP}/\mathsf{poly}$.

Proof. For the first inclusion, let $L \in \mathsf{NP}/\mathsf{log}$. 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 $L(M_i) \subseteq L$. □

In fact, the inclusions in Proposition 3 are proper as we will show in Theorem 5 below. For the proof we need the notion of nondeterministic decision complexity of a string.

Definition 4 (Buhrman, Fortnow, Laplante [5]). *For a time bound t, the* nondeterministic decision complexity of x *is defined as*

$$CND^t(x) = \min\{|M| : M \text{ is a } t\text{-time-bounded nondeterministic Turing machine with } L(M) = \{x\}\ \}\ .$$

As already noted in [1], 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 $CND(x)$ denotes the time-unbounded nondeterministic decision complexity of x.

Inspired by a similar result in [12], we now prove the following separations:

Theorem 5. *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 $CND(x) \geq |x|$. Obviously, $A \in \mathsf{P}/\mathsf{lin}$ 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 $x \in A$, we get $CND(x) \leq c' \log |x| + c'$, contradicting the choice of A. □

From Theorem 5 we infer that both inclusions in Proposition 3 are strict:

Corollary 6. $\mathsf{NP}/\mathsf{log} \subsetneq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subsetneq \mathsf{NP}/\mathsf{poly}$.

4 Proof Systems with Advice

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 [7] and [3].

Definition 7. *Let $k : \mathbb{N} \to \mathbb{N}$ be a function on natural numbers. We say that a proof system f for L* uses k bits of advice*, abbreviated f is a ps/k for L, if there exists 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. *$f(\pi)$ is computable in polynomial time with advice $h(|\ell(\pi)|)$, i.e., for some fixed polynomial-time computable function g, $f(\pi) = g(\pi, h(|\ell(\pi)|))$, and*
3. *for all π, the length of the advice $h(|\ell(\pi)|)$ is bounded by $k(|\pi|)$.*

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 $\ell(\pi) = 1^{|f(\pi)|}$, then f is said to *use k bits of output advice*. Note that the latter notion is only well-defined if we assume that the length of the output $f(\pi)$ (in case $f(\pi)$ is defined) does not depend on the advice.

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 [3]. In [7] and [3], 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 8. *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 $f'(\pi')$ is undefined. 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 input on which f is undefined. □

Cook and Krajíček [7] showed, that TAUT has a $ps/1$ with input advice which p-simulates every ps/log for TAUT, where the p-simulation is computed by a polynomial-time algorithm using $O(\log n)$ bits of advice. The proof of this result easily generalizes to arbitrary languages L, thus yielding:

Theorem 9. *For every language L there exists a proof system P with 1 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 $\pi \mapsto \langle \pi, 1^T, 1^{h(|\ell(\pi)|)}, 1^{p(|\pi|)} \rangle$. □

In contrast, it seems unlikely that a similar result holds for output advice (cf. [3] where we investigated this problem for propositional proof systems).

5 Polynomially Bounded Proof Systems with Advice

In this section we investigate the question whether there exist polynomially bounded proof systems with advice for arbitrary languages L. 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 [8], we obtain the following equivalence:

Theorem 10. *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 9, input advice appears to be a stronger concept than output advice. Surprisingly, the advantage of input advice seems to vanish when we allow a polynomial amount of advice.

Theorem 11. *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 8.

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 10 and 11, the existence of polynomially bounded ps/poly with input advice for L is equivalent to $L \in \mathsf{NP}/\mathsf{poly}$. 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 $\mathsf{NP}/\mathsf{poly}$ is replaced by the instance complexity class $\mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$.

Theorem 12. *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* $\Rightarrow$ *2* follows by definition.

To prove the implication *2* $\Rightarrow$ *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* $\Rightarrow$ *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 $\langle \cdot, \ldots, \cdot \rangle$ 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.

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 $M_x(x)$. □

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

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 3, we get the implications A1 $\Rightarrow$ A2 $\Rightarrow$ A3 for every L. Moreover, by Corollary 6, the inclusions $\mathsf{NP}/\mathsf{log} \subsetneq \mathsf{NIC}[\mathsf{log}, \mathsf{poly}] \subsetneq \mathsf{NP}/\mathsf{poly}$ are proper. Hence we obtain:

Corollary 14. *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.*

6 Polynomially Bounded Proof Systems for TAUT

From a practical point of view, it is most interesting to investigate what precisely happens for $L = \text{TAUT}$ (or more generally for problems in coNP). Even though by Corollary 6, $\mathsf{NP/log}$ and $\mathsf{NIC[log, poly]}$ are distinct, they do not differ inside coNP, as the next theorem shows.

Theorem 15. *Let $L \in \mathsf{coNP}$. Then $L \in \mathsf{NP/log}$ if and only if $L \in \mathsf{NIC[log, poly]}$. Moreover, if $L \in \mathsf{NP/log}$, then the advice can be computed in $\mathsf{FP}^{\mathsf{NP[log]}}$.*

Proof. By Proposition 3 we only have to prove the backward implication. For this let L be a language from coNP. Assuming $L \in \mathsf{NIC[log, poly]}$, 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 $c \log n + c$ 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 $M_1, \ldots, M_{a_n} \in \Pi^n$ and strings $x_1, \ldots, x_{a_n} \in \Sigma^{\leq n}$. Next, N verifies that $x_1, \ldots, x_{a_n}$ do not belong to L. As $L \in \mathsf{coNP}$, this can be done in nondeterministic polynomial time. Then N checks whether for each $i = 1, \ldots, a_n$ 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 $\Pi^n \setminus \{M_1, \ldots, M_{a_n}\}$ is consistent with $L \cap \Sigma^{\leq n}$. After this verification has successfully taken place, N simulates all remaining machines $M \in \Pi^n \setminus \{M_1, \ldots, M_{a_n}\}$ 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 $L \in \mathsf{NIC[log, poly]}$ 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 $L \in \mathsf{NP/log}$, 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 $\mathsf{FP}^{\mathsf{NP[log]}}$. □

By Theorem 11 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 15 we obtain the same equivalence for logarithmic advice.

Corollary 16. 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 [4]:

Theorem 17 (Buhrman, Chang, Fortnow [4]). *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 obtain:

Proposition 18. *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* $\mathsf{PH} \subseteq \mathsf{D}^{\mathsf{p}}$.

Proof. If the polynomial hierarchy collapses to the Boolean hierarchy, then PH in fact collapses to some level BH_k of BH. By Theorem 17, this means that $\mathsf{coNP} \subseteq \mathsf{NP}/k'$ for some constant k'. Hence by Theorem 10, TAUT has a polynomially bounded ps/k' P with output advice. By Theorem 9, 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 10 this implies $\mathsf{coNP} \subseteq \mathsf{NP}/1$ and therefore $\mathsf{PH} \subseteq \mathsf{D}^{\mathsf{p}}$ by Theorem 17. □

So far we have provided different characterizations of the question 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 the following table.

Assumption *if* TAUT *has a polynomially bounded ...*		**Consequence** *then* PH *collapses to ...*
ps/poly	(input or output advice)	$\mathsf{S}_2^{\mathsf{NP}} \subseteq \Sigma_3^{\mathsf{p}}$
ps/log	(input or output advice)	$\mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$
$ps/O(1)$	(input advice)	$\mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$
$ps/O(1)$	(output advice)	$\mathsf{P}^{\mathsf{NP}[O(1)]} = \mathsf{BH}$
$ps/0$	(no advice)	NP

The first line in the above table follows from Theorems 10 and 11 and a result of Cai, Chakaravarthy, Hemaspaandra, and Ogihara [6], who have shown that $\mathsf{coNP} \subseteq \mathsf{NP}/\mathsf{poly}$ implies $\mathsf{PH} \subseteq \mathsf{S}_2^{\mathsf{NP}}$.

For the second line, the distinction between input and output advice is again irrelevant (Corollary 16). Here we use a result of Arvind, Köbler, Mundhenk, and Torán [1], who showed that TAUT $\in \mathsf{NIC}[\mathsf{log}, \mathsf{poly}]$ implies $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}[\mathsf{log}]}$.

Finally, the constant-advice case (lines 3 and 4), follows from Theorem 17 in conjunction with Theorems 10 and 12. 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).

References

1. V. Arvind, J. Köbler, M. Mundhenk, and J. 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.

2. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin Heidelberg, 1988.
3. O. Beyersdorff and S. Müller. A tight Karp-Lipton collapse result in bounded arithmetic. In *Proc. 17th Workshop on Computer Science Logic*, volume 5213 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 2008.
4. H. Buhrman, R. Chang, and L. Fortnow. One bit of advice. In *Proc. 20th Symposium on Theoretical Aspects of Computer Science*, pages 547–558, 2003.
5. H. Buhrman, L. Fortnow, and S. Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2001.
6. J.-Y. Cai, V. T. Chakaravarthy, L. A. Hemaspaandra, and M. Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.
7. S. A. Cook and J. Krajíček. Consequences of the provability of $NP \subseteq P/poly$. *The Journal of Symbolic Logic*, 72(4):1353–1371, 2007.
8. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.
9. R. M. Karp and R. 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.
10. J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54:1063–1079, 1989.
11. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, Berlin Heidelberg, 2. edition, 1997.
12. P. Orponen, K. Ko, U. Schöning, and O. Watanabe. Instance complexity. *Journal of the ACM*, 41(1):96–121, 1994.
13. P. Pudlák. The lengths of proofs. In S. R. Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, Amsterdam, 1998.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/