ACADEMIC PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Journal of Computer and System Sciences 67 (2003) 608–622

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

One-way permutations and self-witnessing languages ☆

Christopher M. Homan*,[1] and Mayur Thakur

Department of Computer Science, University of Rochester, Rochester, NY 14627, USA

Received 23 May 2002; revised 28 March 2003

Abstract

A desirable property of one-way functions is that they be total, one-to-one, and onto—in other words, that they be permutations. We prove that one-way permutations exist exactly if $P \neq UP \cap coUP$. This provides the first characterization of the existence of one-way permutations based on a complexity-class separation and shows that their existence is equivalent to a number of previously studied complexity-theoretic hypotheses. We study permutations in the context of witness functions of nondeterministic Turing machines. A language is in PermUP if, relative to some unambiguous, nondeterministic, polynomial-time Turing machine accepting the language, the function mapping each string to its unique witness is a permutation of the members of the language. We show that, under standard complexity-theoretic assumptions, PermUP is a strict subset of UP. We study SelfNP, the set of all languages such that, relative to some nondeterministic, polynomial-time Turing machine that accepts the language, the set of all witnesses of strings in the language is identical to the language itself. We show that $SAT \in SelfNP$, and, under standard complexity-theoretic assumptions, $SelfNP \neq NP$.

Keywords: One-way functions; Permutations; One-to-one functions; Complexity-theoretic cryptography; Self-witnessing languages

1. Introduction

Until the results in this paper were known, the question "What complexity class separations, if any, characterize the existence of one-way permutations (i.e., total, one-to-one, onto, one-way functions)?" has remained open. Throughout this paper, when we say "one-way functions" we mean "worst-case one-way functions" as opposed to "average-case one-way functions."

☆ A preliminary version of this paper was presented at the 2nd IFIP TCS conference.

* Corresponding author.

E-mail addresses: choman@cs.rochester.edu (C.M. Homan), thakur@cs.rochester.edu (M. Thakur).

[1] Supported in part by Dept. of Education (GAANN program) grant EIA-0080124, by grant NSF-INT-9815095/DAAD-315-PPP-gü-ab, and by grant NSF-CCR-9322513.

0022-0000/03/$ - see front matter
doi:10.1016/S0022-0000(03)00068-0

Average-case one-way functions have been studied extensively [9,18,21,24,28] and are used widely in applied cryptography. We prove that one-way permutations exist exactly if $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$. Recall [26] that UP is the class of all languages accepted by a nondeterministic Turing machine that runs in polynomial time and has, on any input, at most one accepting path. Such Turing machines are called "UPTMs," or *unambiguous*, *polynomial-time Turing machines*.

Grollmann and Selman [11] and, independently, Ko [19] (see also the work by Berman [4]) show that $\mathrm{P} \neq \mathrm{UP}$ exactly if total, one-to-one, (but not necessarily onto) one-way functions exist and that $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$ exactly if *partial*, one-to-one, onto, one-way functions exist. In this paper, we extend their results to *total*, one-to-one, *onto*, one-way functions. The existence of one-way permutations is thus equivalent to a number of hypotheses [7,10–12,15,19,23], including the following.

(1) The weak definability principle does not hold for some logic that is closed under first-order operations [10].
(2) $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{P}$ [23].
(3) There exist UPTMs M and N such that $L(M) \subseteq L(N)$ and $f \notin \mathrm{FP}$ for all functions f having the property that, for all $x \in L(M)$, $f(\langle x, \mathrm{wit}_M(x) \rangle) = \mathrm{wit}_N(x)$ [7].

$\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$, introduced by Hemaspaandra et al. [16] (see also [23]), is the class of all languages L in UP for which the following holds for all unambiguous polynomial-time Turing machines U accepting L: the mapping between a member of L and its unique witness relative to U (i.e., the bits guessed by a nondeterministic accepting path of U) is polynomial-time computable. By definition, $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \subseteq \mathrm{P}$. Item 3 above follows by analogy to a result by Fenner et al. [7], who show that onto, one-way functions do not exist exactly if all nondeterministic polynomial-time Turing machines accepting a language have roughly (i.e., modulo a polynomial-time transformation) the same witnesses. Theorem 3.4, stated in Section 3 of this paper, lists all hypotheses known to be equivalent to the existence of one-way permutations.

We also study permutations in the context of witness functions of nondeterministic Turing machines. In this context, a function f is a *permutation* of a language L if f is a (partial) one-to-one function defined over exactly the members of L such that the image of f is exactly L. We say a function *permutes* a language if the function is a permutation of the language. Let PermUP, or *self-permuting* UP, be the set of all languages such that there is a UPTM accepting the language, where the mapping between each string in the language and the unique witness used by the UPTM to accept the string permutes the language. That is, $\mathrm{PermUP} = \{L \mid \text{there exists a UPTM } U \text{ such that } L(U) = L \text{ and } \mathrm{wit}_U \text{ permutes } L\}$, where wit_U denotes a function that maps each $x \in L$ to its unique witness in U.

By definition, $\mathrm{PermUP} \subseteq \mathrm{UP}$. It is easy to see that any language $L \in \mathrm{P}$ is in PermUP via the following "simple UPTM" (i.e., a UPTM whose witness function wit_U is computable in polynomial-time).

> On input x nondeterministically guess a string y of length exactly $|x|$ and accept if and only if $x \in L$, and $y = x$.

One might wonder if the restriction imposed on PermUP machines, namely that they have a witness scheme that is a permutation, makes PermUP trivially simple. In particular, one might

wonder if PermUP is contained in P. We show, however, that the closure of PermUP under polynomial-time, one-to-one reductions is UP. Thus, PermUP captures the most complex languages in UP. In other words, assuming $P \neq UP$, some languages in PermUP are accepted via "complex UPTMs" (i.e., UPTMs whose witness functions are not polynomial-time computable).

On the other hand, we show that it is unlikely that all languages in UP are in PermUP (i.e., that PermUP is closed under polynomial-time, many-one reductions), for, if this is the case, then $E = UE$. Thus, under standard complexity-theoretic assumptions, PermUP is a nontrivial subset of UP that captures the hardest languages in UP.

It appears, then, that PermUP contains languages that are simple (i.e., accepted by some simple UPTM) and it contains languages that are complex (i.e., accepted by some complex UPTM). But is there a language in PermUP that is accepted by both a simple and a complex UPTM? The answer to this question is linked to the existence of one-way permutations. Consider the class $\text{HiPermUP} = \{L \mid \text{there exists a UPTM } U \text{ such that } L = L(U), \text{wit}_U \text{ permutes } L, \text{and wit}_U \text{ is not polynomial-time computable}\}$, which is a (possibly empty) subset of PermUP. By definition, languages in HiPermUP are in PermUP via a complex UPTM. We show that one-way permutations exist exactly if there is a language $L \in P \cap \text{HiPermUP}$. Such an L is thus in PermUP via both a simple and a complex UPTM.

We also study the class SelfNP, or *self-witnessing* NP, which we regard as the natural NP analog of PermUP. SelfNP is defined as $\{L \mid \text{there exists a nondeterministic Turing machine } N \text{ that runs in polynomial time such that } L(N) = L \text{ and } \bigcup_{x \in L} \text{wit}_N(x) = L\}$, where wit_N is a function that maps each $x \in L$ to its set of witnesses relative to N. We show that the relationship between SelfNP and NP is basically analogous to the relationship between PermUP and UP (i.e., the closure of SelfNP under polynomial-time many-one reductions is NP, and if $\text{SelfNP} = \text{NP}$, then $E = NE$). Furthermore, $\text{SAT} \in \text{SelfNP}$.

The remainder of the paper is organized as follows. Section 2 presents definitions and notations. Section 3 proves results on the existence of one-way permutations. Section 4 proves results related to PermUP and SelfNP. Section 5 concludes the paper and presents future directions.

2. Definitions and notations

All sets in this paper, unless otherwise stated, are subsets of Σ^*, where Σ is the standard alphabet $\{0, 1\}$. The length of a string x is denoted by $|x|$. For any set S, $||S||$ denotes the cardinality of S. For any string $w \in \Sigma^*$ and any set $S \subseteq \Sigma^*$, $wS = \{ws \mid s \in S\}$. For any two sets $S, T \subseteq \Sigma^*$, $ST = \{uv \mid u \in S \wedge v \in T\}$.

For each Turing machine N and each $x \in \Sigma^*$, $N(x)$ means "the computation of N on input x." NPTM (respectively, NETM) means "nondeterministic polynomial-time (respectively, exponential-time) Turing machine." P is the class of all polynomial-time computable languages. E is the class of all languages accepted by some deterministic exponential-time Turing machine. NP (respectively, NE) denotes the class of all languages L such that L is accepted by an NPTM (respectively, NETM). UPTM (respectively, UETM) means "unambiguous polynomial-time (respectively, exponential-time) Turing machine," that is, N is a UPTM if and only if N is an NPTM and, on any input $x \in \Sigma^*$, N has at most one accepting path. UP (respectively, UE) is the class of all languages L such that L is accepted by some UPTM (respectively, UETM). UP was

defined by Valiant in [26] and UE was defined and studied in [14,22]. TALLY is the class of all tally languages, that is, $\mathrm{TALLY} = \{L \mid L \subseteq 1^*\}$. The set of total, polynomial-time computable functions is called FP.

We say that $A \leqslant^p_m B$ (*A polynomial-time many-one reduces to B*) if there exists a total, polynomial-time computable function σ such that, for each $x \in \Sigma^*$, $x \in A$ if and only if $\sigma(x) \in B$. We say that $A \leqslant^p_{1\text{-}1} B$ (*A polynomial-time one-to-one reduces to B*) if there exists a one-to-one function $\sigma \in \mathrm{FP}$ such that $A \leqslant^p_m B$ via σ. For each complexity class $\mathscr{C}$ and all a, b such that the reducibility $\leqslant^a_b$ is defined, the *reduction closure* of $\mathscr{C}$ relative to $\leqslant^a_b$, denoted $\mathrm{R}^a_b(\mathscr{C})$, is the set $\{L \subseteq \Sigma^* \mid (\exists L' \in \mathscr{C})[L \leqslant^a_b L']\}$.

For each NPTM N and each $x \in L(N)$, a string y is said to be a *witness* for x relative to N if there is an accepting path of $N(x)$ on which the sequence of nondeterministic guess bits is exactly y. Note that y (unless by coincidence) does not necessarily encode x, i.e., it is *naked*. Denote the mapping from each $x \in L(N)$ to the set of witnesses for x relative to N as $\mathrm{wit}_N : \Sigma^* \to 2^{\Sigma^*}$ (where 2^{Σ^*} is the set of all subsets of Σ^*). When N is a UPTM, since each $x \in \Sigma^*$ has at most one witness relative to such an N, for convenience we regard wit_N as a (partial) mapping from Σ^* to Σ^*.

For each function f let $\mathrm{im}(f)$ denote the image of f. For each function $f : D \to R$ and each $S \subseteq D$ such that f is defined on each element in S, let $f(S)$ denote the set $\{r \in R \mid (\exists s \in S)[r = f(s)]\}$.

For any set $S \subseteq \Sigma^*$, a function $f : \Sigma^* \to \Sigma^*$ is a *permutation* of S if the set of all strings in Σ^* on which f is defined is exactly S, $\mathrm{im}(f) = S$, and f is one-to-one. We say f *permutes* S if f is a permutation of S.

A function $f : \Sigma^* \to \Sigma^*$ is *polynomial-time invertible* (alternatively, FP-*time invertible*) if there is a function $g \in \mathrm{FP}$ such that for every $y \in \mathrm{im}(f)$, $f(g(y)) = y$. Let $\langle \cdot, \cdot \rangle$ denote a standard, fixed, polynomial-time computable, polynomial-time invertible, total, one-to-one, onto function from $\Sigma^* \times \Sigma^*$ to Σ^* such that the output of the function is strictly length increasing in the lengths of either of its arguments when the other argument is fixed. Such a function is called a *pairing function*.

Definition 2.1 (Grollmann and Selman [11], Ko [19], Berman [4], Selman [25]). A function $f : \Sigma^* \to \Sigma^*$ is *honest* if there exists a polynomial p (called the *honesty polynomial*) such that for all $y \in \mathrm{im}(f)$ there exists an $x \in \Sigma^*$ such that $f(x)$ is defined, $y = f(x)$, and $|x| \leqslant p(|y|)$.

Intuitively, if an honest function is hard to invert, then it is so "honestly," i.e., not merely because the function shrinks the input too much. Grollmann and Selman [11] provided the first independent study of complexity-theoretic, (one-to-one) one-way functions (see also [4,19]). The definition below is for complexity-theoretic, one-way functions of arbitrary ambiguity [27].

Definition 2.2 (Watanabe [27], see [4,11,19,25]). A function $f : \Sigma^* \to \Sigma^*$ is *one-way* if f is honest, polynomial-time computable, and not polynomial-time invertible.

Definition 2.3 (see [17,27]). For $g : \mathbb{N} \to \mathbb{N}$, we say a function $\sigma : \Sigma^* \to \Sigma^*$ is g-to-1 if

$$(\forall y \in \mathrm{im}(\sigma))[||\{x \in \Sigma^* \mid \sigma(x) = y\}|| \leqslant g(|y|)].$$

3. One-way permutations

In this section we prove the main result of this paper.

Theorem 3.1. $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$ *if and only if one-way permutations exist.*

As noted in the introduction, it is known (see [4,11,19]) that $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$ exactly if partial, one-to-one, onto, one-way functions exist. The independent existence of both partial, one-to-one, onto, one-way functions and one-way permutations has been studied in a variety of settings (see [7,10–12,15,19,23]). Theorem 3.2 below collects results either previously known to be equivalent to the existence of partial, one-to-one, onto, one-way functions or that can be obtained by arguments analogous to previously known proofs. In particular, the equivalence of items 1 and 2 is due to Ko [19], of items 1–4 to Grollmann and Selman [11], of items 2 and 5 to Hartmanis and Hemachandra [12], of items 3 and 5–8 to Rothe and Hemaspaandra [23], and of 3 and 9 to Grädel [10]. The equivalence of 1, 2, 5, 10, and 11 is analogous to results for the existence of onto, one-way functions by Fenner et al. [7].

Theorem 3.2. *The following are equivalent.*

(1) *There exists a partial, one-to-one, onto, one-way function.*
(2) $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$.
(3) *There exists a partial, one-to-one, one-way function* f *such that* $\mathrm{im}(f) \in \mathrm{P}$.
(4) *There exists a total, one-to-one, one-way function* f *such that* $\mathrm{im}(f) \in \mathrm{P}$.
(5) $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \neq \mathrm{P}$.
(6) $1\text{-}\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \neq \mathrm{P}$.
(7) $\Sigma^* \notin \mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$.
(8) $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$ *is not closed under complementation.*
(9) *The weak definability principle does not hold for some logic that is closed under first order operations.*
(10) $\mathrm{UPSV_t} \not\subseteq \mathrm{FP}$.
(11) *There exist* UPTM*s* M *and* N *such that* $L(M) \subseteq L(N)$ *and* $f \notin \mathrm{FP}$ *for all functions* f *having the property that, for all* $x \in L(M)$, $f(\langle x, \mathrm{wit}_M(x) \rangle) = \mathrm{wit}_N(x)$.

Rothe and Hemaspaandra [23] define $1\text{-}\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$ as the set of all languages L in UP for which the following holds for each UPTM U accepting L: there exists a polynomial-time computable function f_U such that, for all $x \in L$, $f_U(x)$ outputs the first bit of the accepting path of $U(x)$. Fenner et al. [7] consider the "NP version" of $1\text{-}\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$, however their results regarding this "NP version" [7] are not known to be analogous to the above result for $1\text{-}\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP})$.

Regarding the existence of one-way permutations, the following equivalence, due to Hemaspaandra and Rothe, is known. Note that a set L is P-rankable [8] if there exists a

polynomial-time computable function r_L such that, for all $x \in \Sigma^*$, $r_L(x)$ is the rank of x in L (i.e., the number of strings in L that are lexicographically less than or equal to x).

Theorem 3.3 ([15]). *The following are equivalent.*

(1) *There exists a one-way permutation.*
(2) *There exists a total, one-to-one, one-way function whose range is* P*-rankable.*

As a result of Theorem 3.1, we have the following.

Theorem 3.4. *The following are equivalent.*

(1) *Any item from Theorem* 3.2.
(2) *Any item from Theorem* 3.3.
(3) $\Sigma^* \in \mathrm{HiPermUP}$.
(4) *There exists* $L \in \mathrm{P}$ *such that* $L \in \mathrm{HiPermUP}$.

We defer the proof of Theorem 3.4 until the end of this section. We now prove Theorem 3.1.

Proof of Theorem 3.1. The proof follows immediately from the equivalence of items 1 and 2 of Theorem 3.2 and the application of Lemma 3.5 for $g(n) = 1$.

Lemma 3.5. *Let g be a nondecreasing function from* $\mathbb{N}$ *to* $\mathbb{N}^+$. *There exists a partial, g-to-*1, *onto, one-way function if and only if there exists a total, g-to-*1, *onto, one-way function.*

Proof. The ($\Leftarrow$) direction is easy, since all total functions are partial functions. For the ($\Rightarrow$) direction, let h be a partial, g-to-1, onto, one-way function for some nondecreasing $g : \mathbb{N} \to \mathbb{N}^+$. We claim that f, defined on input x as

$$f(x) = \begin{cases} 1^n 0 y & \text{if } (x = 1^{n+1}0y) \wedge (h(y) \text{ is defined}) \wedge (n > 0), \\ 01h(y) & \text{if } (x = 10y) \wedge (h(y) \text{ is defined}), \\ 0^{n+1}1y & \text{if } (x = 0^n 1y) \wedge (n > 0), \\ x & \text{otherwise,} \end{cases}$$

is total, g-to-1, onto, and one-way.

The intuition behind the proof is captured in Figs. 1 and 2. Fig. 1 is a graphical representation of an example of h, where the vertices of the graph are the strings in Σ^* and the edges are the relations defined by h, i.e., there is an edge from x to y if and only if $h(x)$ is defined and $h(x) = y$. (Note that the predicate "$h(x)$ is defined" is polynomial-time computable. This is because h can be computed in polynomial time by, for instance, a clocked Turing machine.) It is easy to see that a function is total precisely when the outdegree of every vertex in the graph representing the function is exactly one and is onto whenever the indegree of every vertex is at least one. Since h is

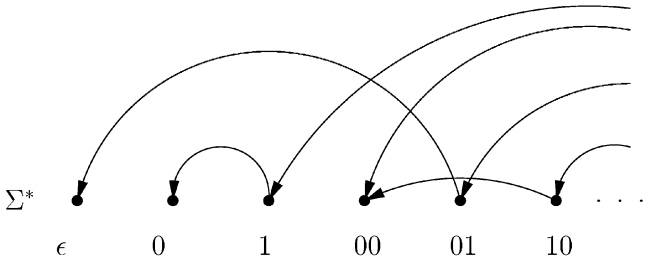

Fig. 1. A graph where the vertices are the elements of Σ^*, and the edges are the mapping defined by h.

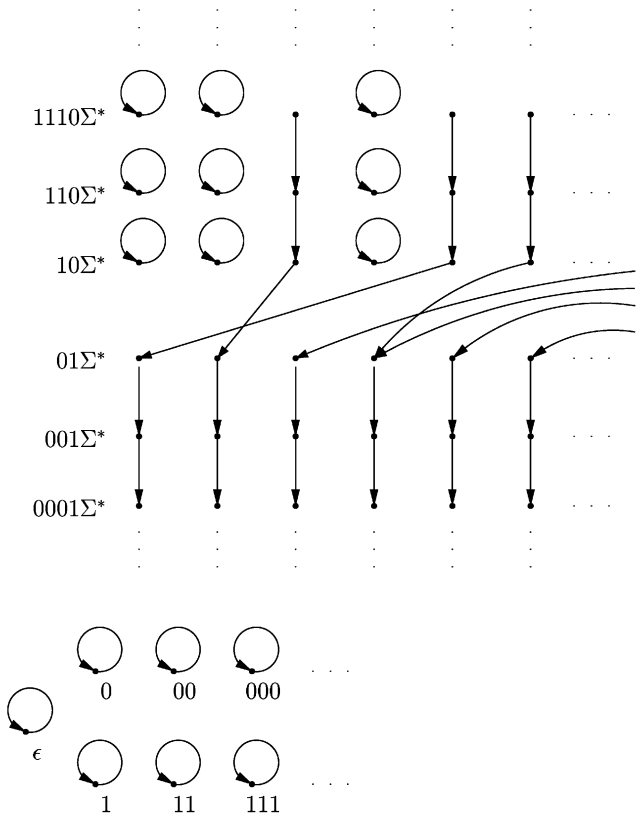

Fig. 2. A graph where the vertices are the elements of Σ^*, and the edges are the mapping defined by f.

not necessarily total, each vertex of the graph in Fig. 1 has an outdegree of either zero or one, however since h is onto, every vertex has an indegree of at least one.

The construction of f essentially imposes a two-dimensional structure on Σ^*, as shown in Fig. 2, and embeds h into this structure by identifying the domain of h with the elements in row $10\Sigma^*$ and the image of h with the elements in row $01\Sigma^*$. The construction then fills the graph in with additional edges in a way that guarantees that "one-way"-ness is preserved and that every vertex x

has an outdegree of exactly one and an indegree of at least one and at most $g(|x|)$ (since the indegree represents the "many-to-one"-ness of the function). By checking that the graph in Fig. 2 has these properties, it is easy to see that f is a total, g-to-one, onto, one-way function.

Continuing with the formal proof, it is easy to see that f is total and polynomial-time computable. To see that f is onto, note that

$$f(11^*0\{y \mid h(y) \text{ is defined}\}) = 11^*0\{y \mid h(y) \text{ is defined}\} \cup 01\Sigma^*,$$

by the first two conditions in the definition of f, and because h is onto. Furthermore,

$$f(00^*1\Sigma^*) = 000^*1\Sigma^*,$$

by the third condition in the definition of f, and

$$f((\Sigma^* - 11^*0\{y \mid h(y) \text{ is defined}\}) - 00^*1\Sigma^*) = (\Sigma^* - 11^*0\{y \mid h(y) \text{ is defined}\}) - 00^*1\Sigma^*,$$

by the last condition in the definition of f, so clearly, $\mathrm{im}(f) = \Sigma^*$.

To see that $f(x)$ is g-to-1, choose an arbitrary $z \in \Sigma^*$. If for some $z' \in \Sigma^*$, $z = 01z'$, then $||f^{-1}(z)|| = ||h^{-1}(z')||$. Since g is nondecreasing, $||h^{-1}(z')|| \leqslant g(|z'|) \leqslant g(|z|)$. If $z \notin 01\Sigma^*$ then by the definition of f, $||f^{-1}(z)|| = 1$, thus f is g-to-1.

To see that f is one-way, first note that f is honest. Next, suppose that there exists a function $\gamma \in \mathrm{FP}$ that inverts f. We could then invert h in polynomial time as follows.

On input y, compute $\gamma(01y) = 10w$ and output w.

We thus conclude that f is total, g-to-1, onto, and one-way. □

We now prove Theorem 3.4.

Proof of Theorem 3.4. The equivalence of 1 and 2 follows immediately from Theorem 3.1.

To prove 2 ⇒ 3, we show that Theorem 3.3, item 1 implies that $\Sigma^* \in \mathrm{HiPermUP}$. Suppose that f is a one-way permutation with honesty polynomial p. Let N be a nondeterministic Turing machine defined as follows.

On input x, nondeterministically guess a string y of length at most $p(|x|)$, and accept if and only if $f(y) = x$.

Clearly, $L(N) = \Sigma^*$, and N is a UPTM. Also, wit_N, which is equivalent to f^{-1}, permutes Σ^* and is not polynomial-time computable (since otherwise f would be polynomial-time invertible). Thus $\Sigma^* \in \mathrm{HiPermUP}$.

To prove 3 ⇒ 1, we show that $\Sigma^* \in \mathrm{HiPermUP}$ implies Theorem 3.2, item 2. Suppose $\Sigma^* \in \mathrm{HiPermUP}$. Let U be a UPTM such that $L(U) = \Sigma^*$, wit_U is a permutation of Σ^*, and wit_U is not polynomial-time computable. Then the language $L = \{\langle x, y\rangle \mid y \text{ is lexicographically less than } \mathrm{wit}_U(x)\}$ is in $\mathrm{UP} \cap \mathrm{coUP}$. Also $L \notin \mathrm{P}$, since if $L \in \mathrm{P}$, then we could compute $\mathrm{wit}_U(x)$ in polynomial time via binary search. Thus, Theorem 3.2, item 2 holds.

Clearly, 3 ⇒ 4. To prove 4 ⇒ 3, let $L \in \mathrm{P} \cap \mathrm{HiPermUP}$. Let N be a UPTM such that N accepts L, wit_N permutes L, and wit_N is not polynomial-time computable. Let p be a polynomial that bounds the running time of N. Then the following NPTM U accepts Σ^*.

On input x, guess a string y of length at most $\max\{|x|, p(|x|)\}$. If $x \in L$ and $y = \mathrm{wit}_N(x)$, then accept. If $x \notin L$ and $y = x$, then accept. Otherwise, reject.

Clearly, U is a UPTM, and wit_U permutes Σ^*. Furthermore, wit_U is not polynomial-time computable, since otherwise we could use it to compute wit_N in polynomial time. Thus $\Sigma^* \in \mathrm{HiPermUP}$, and $3 \Leftrightarrow 4$. □

4. Self-witnessing languages

In this section, we study properties of PermUP and SelfNP. First, we show that the closure of PermUP under polynomial-time, one-to-one reductions is UP.

Theorem 4.1. $\mathrm{UP} = \mathrm{R}^p_{1\text{-}1}(\mathrm{PermUP})$.

Proof. Let U be a UPTM whose running time is bounded by some nondecreasing polynomial q. Let L' be a language defined as follows.

$$L' = \{\langle x, x\rangle \mid U \text{ accepts } x\} \cup \{\langle x, y\rangle \mid y = \mathrm{wit}_U(x)\}.$$

We claim that $L' \in \mathrm{PermUP}$. Consider the following NPTM U', which accepts L'.

On input $\langle x, z\rangle$ nondeterministically guess a string $\langle x, y\rangle$, where $|y| \leqslant \max\{|x|, q(|x|)\}$. If $z = x$ and $y = \mathrm{wit}_U(x)$, then accept. If $z = \mathrm{wit}_U(x)$ and $y = x$, then accept. Otherwise, reject.

Clearly, U' is a UPTM. The following function $r : \Sigma^* \to \Sigma^*$, defined on input x below, is a polynomial-time, one-to-one reduction from $L(U)$ to L'.

$$r(x) = \langle x, x\rangle.$$

To prove that $L' \in \mathrm{PermUP}$, note that, for all $x \in L(U)$ and the unique y such that $y = \mathrm{wit}_U(x)$, it holds that $\mathrm{wit}_{U'}(\langle x, x\rangle) = \langle x, y\rangle$ and $\mathrm{wit}_{U'}(\langle x, y\rangle) = \langle x, x\rangle$. It follows that $\mathrm{wit}_{U'}$ is a permutation of L'. □

As an immediate corollary to Theorem 4.1, we get the following.

Corollary 4.2. $\mathrm{P} \neq \mathrm{UP} \Leftrightarrow \mathrm{P} \neq \mathrm{PermUP}$.

The following theorem shows that asking whether $\mathrm{P} = \mathrm{PermUP} \cap \mathrm{coPermUP}$ is logically the same as asking whether $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$, which, by Theorem 3.1, is equivalent to asking whether one-way permutations exist.

Theorem 4.3. $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP} \Leftrightarrow \mathrm{P} \neq \mathrm{PermUP} \cap \mathrm{coPermUP}$.

Proof. The ($\Leftarrow$) direction is easy. For the ($\Rightarrow$) direction, suppose $L \in \mathrm{UP} - P$ and $\bar{L} \in \mathrm{UP}$. Let U be a UPTM accepting L whose running time is bounded by some nondecreasing polynomial q, and let L' be as in the proof of Theorem 4.1. Clearly, $L' \in \mathrm{PermUP} - \mathrm{P}$. Let U_1 be a

UPTM accepting $\bar{L}$ whose running time is bounded by some nondecreasing polynomial p. Let S be $\{\langle x,x\rangle \mid U_1 \text{ accepts } x\} \cup \{\langle x,y\rangle \mid y = \text{wit}_{U_1}(x)\} \cup \{\langle x,y\rangle \mid (x \neq y) \wedge (y \neq \text{wit}_U(x)) \wedge (y \neq \text{wit}_{U_1}(x))\}$. It is easy to check that $L' \cup S = \Sigma^*$ and that $L' \cap S = \emptyset$. The following NPTM N accepts S.

> On input $\langle x,z\rangle$, nondeterministically guess $\langle x,y\rangle$, where $|y| \leqslant \max\{p(|x|), q(|x|), |x|\}$. If $z = x$ and $y = \text{wit}_{U_1}(x)$, then accept. If $z = \text{wit}_{U_1}(x)$ and $y = x$, then accept. If $z \neq x$, $y = z$, $y \neq \text{wit}_U(x)$, and $y \neq \text{wit}_{U_1}(x)$, then accept. Otherwise, reject.

Clearly, N is a UPTM. It is easy to see that wit_N permutes S. Thus, since $\bar{L}' = S$, we conclude that $\bar{L}' \in \text{PermUP}$, and $L' \in (\text{PermUP} \cap \text{coPermUP}) - \text{P}$. □

We wish to define a natural relaxation of PermUP that would include languages in NP − UP (if indeed UP ≠ NP). One important distinction between UPTMs and NPTMs is that the witness functions of UPTMs are single-valued (because there is at most one accepting path), but those of NPTMs may be multivalued (since there can be more than one accepting path). SelfNP, as defined in the introduction, is a natural NP analog of PermUP, where instead of requiring the witness function to be a permutation, we only require that the set of witnesses is the same as the language. Note that the "self-witnessing property," i.e., the property that witnesses themselves be part of the language, also holds for languages in PermUP, since for these languages the witness function is a permutation of the language.

Theorem 4.4 shows that SAT is a member of SelfNP.

Theorem 4.4. $\text{SAT} \in \text{SelfNP}$.

In order to prove Theorem 4.4, consider the following definitions. We say an NPTM M is *self-contained witnessing* if $\bigcup_{x \in L(M)} \text{wit}_M(x) \subseteq L(M)$. We say an NPTM M is *honest* if there exists a polynomial p such that for all $x \in L(M)$, there exists a $w \in \text{wit}_M(x)$ such that $p(|w|) \geqslant |x|$. Consider the following proposition.

Proposition 4.5. *For each language* $L \in \text{NP}$, *if there exists a self-contained witnessing, honest* NPTM *that accepts* L, *then* $L \in \text{SelfNP}$.

Proof. Choose $L \in \text{NP}$ and suppose that there exists a self-contained witnessing, honest NPTM M that accepts L. Let p be a polynomial witnessing that M is honest. We may assume without loss of generality that p also bounds the running time of N as a function of the input length. Consider the NPTM M', which does the following.

> On input $x \in \Sigma^*$, nondeterministically guess a string y of length at most $p(|x|)$ and accept if and only if $y \in \text{wit}_M(x)$ or $x \in \text{wit}_M(y)$.

Clearly, $L \subseteq L(M')$. Also, note that $L(M') \subseteq L$ because, by construction, any string x that is accepted by M' either is accepted by M too (in which case, $x \in L$, since, by assumption, $L(M) = L$) or is a witness of some string y such that $y \in L$ (in which case, $x \in L$, since M is self-contained witnessing). We next show that $\bigcup_{x \in L} \text{wit}_{M'}(x) = L$. First, choose $w \in \bigcup_{x \in L} \text{wit}_{M'}(x)$. By the

description of M' above, $w \in L$ either directly or because $w \in \bigcup_{x \in L} \mathrm{wit}_M(x)$ and M is self-contained witnessing. Thus, $\bigcup_{x \in L} \mathrm{wit}_{M'}(x) \subseteq L$. Next, choose $w \in L$. Since M is honest, there exists some $z \in \mathrm{wit}_M(w)$ such that $|w| \leqslant p(|z|)$. Thus, by the definition of M', $w \in \mathrm{wit}_{M'}(z)$, from which it follows that $w \in \bigcup_{x \in L} \mathrm{wit}_{M'}(x)$. Thus, $\bigcup_{x \in L} \mathrm{wit}_{M'}(x) = L$. We conclude that $L \in \mathrm{SelfNP}$. □

In light of Proposition 4.5, Theorem 4.4 easily follows from the lemma below.

Lemma 4.6. SAT *is accepted by an honest, self-contained witnessing* NPTM.

Proof. We will construct an NPTM N such that $L(N) = \mathrm{SAT}$, N is honest, and $\bigcup_{x \in \mathrm{SAT}} \mathrm{wit}_N(x) \subseteq \mathrm{SAT}$. Fix some binary encoding of boolean formulas. We assume without loss of generality that the lengths of any two strings that encode formulas having the same number of literals are polynomially related in length. We describe N as follows.

On input $x \in \Sigma^*$, N interprets x as a boolean formula, which we denote as $\phi[X_1, \ldots, X_m]$, where $X_1, \ldots, X_m$ are precisely the variables in ϕ. N nondeterministically guesses a boolean formula ψ of the form $\bigwedge_{j=1}^{c} A_1 \wedge \cdots \wedge A_m$, where c is the number of literals in $\phi[X_1, \ldots, X_m]$, and, for each $1 \leqslant i \leqslant m$, $A_i \in \{X_i, \neg X_i\}$. Thus, ψ has at most c^2 literals, and so the string encoding ψ can be guessed in time polynomial in the length of x. Moreover, the lengths of the strings encoding $\phi[X_1, \ldots, X_m]$ and ψ are polynomially related, thus N is honest.

For all i such that $1 \leqslant i \leqslant m$, let a_i be defined as follows.

$$a_i = \begin{cases} \text{false} & \text{if } A_i = \neg X_i, \\ \text{true} & \text{if } A_i = X_i. \end{cases}$$

N accepts if and only if $\phi[X_1 = a_1, \ldots, X_m = a_m]$ is true, where, for any boolean formula F with variables $Y_1, \ldots, Y_k$, $F[Y_1 = b_1, \ldots, Y_k = b_k]$ represents the boolean formula obtained by setting Y_1 to b_1, Y_2 to b_2, etc. in F. Thus, N accepts SAT. Clearly, each such $\psi \in \mathrm{SAT}$. Thus, N is self-contained witnessing. □

From Theorem 4.4, the corollary below easily follows.

Corollary 4.7. (1) $\mathrm{P} \neq \mathrm{NP} \Leftrightarrow \mathrm{P} \neq \mathrm{SelfNP}$.

(2) $\mathrm{NP} = \mathrm{R}_m^p(\mathrm{SelfNP})$.

(3) $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \Leftrightarrow \mathrm{P} \neq \mathrm{SelfNP} \cap \mathrm{coSelfNP}$.

(4) *For all* $L \subseteq \Sigma^*$, *if there is a polynomial-time computable, honest, onto reduction from* L *to* SAT *then* $L \in \mathrm{SelfNP}$.

(Part 3 above actually follows by analogy to Theorem 4.3.)

Theorem 4.1 and Corollary 4.7, part 2 show that PermUP and SelfNP capture the hardest problems in UP and NP, respectively. Theorem 4.9, which follows from Lemma 4.8 below (both are due to Hemaspaandra [13]), show that it is unlikely that either $\mathrm{SelfNP} = \mathrm{NP}$ or $\mathrm{PermUP} = \mathrm{UP}$.

Lemma 4.8. $\mathrm{TALLY} \cap \mathrm{SelfNP} \subseteq \mathrm{P}$.

Proof. Choose $L \in \mathrm{TALLY} \cap \mathrm{SelfNP}$. Thus, there exists an NPTM N accepting L such that, for all $x \in L$, $\mathrm{wit}_N(x) \subseteq 1^*$. But then, for each x, it is possible to enumerate $\mathrm{wit}_N(x)$ in polynomial time. Thus, $L \in \mathrm{P}$. □

Theorem 4.9. (1) $\mathrm{PermUP} = \mathrm{UP} \Rightarrow \mathrm{E} = \mathrm{UE}$.
(2) $\mathrm{SelfNP} = \mathrm{NP} \Rightarrow \mathrm{E} = \mathrm{NE}$.

Proof. For part 1, suppose that $\mathrm{PermUP} = \mathrm{UP}$. Thus, by Lemma 4.8 and the fact that $\mathrm{PermUP} \subseteq \mathrm{SelfNP}$, it follows that $\mathrm{TALLY} \cap \mathrm{UP} \subseteq \mathrm{P}$. By analogy to Book's Theorem [6], $\mathrm{E} = \mathrm{UE}$. The proof of part 2 follows by analogy to the proof of part 1. □

The oracle result below shows that relativizable techniques cannot collapse the various self-witnessing languages we have studied so far.

Theorem 4.10. *There exists an oracle B such that*

(1) $\mathrm{SelfNP}^B \neq \mathrm{UP}^B$,
(2) $\mathrm{PermUP}^B \neq \mathrm{UP}^B$,
(3) $\mathrm{SelfNP}^B \neq \mathrm{NP}^B$, *and*
(4) $\mathrm{P}^B \neq \mathrm{PermUP}^B$.

Proof. Let B be the oracle constructed in the proof of Baker et al. [1, Theorem 3] that $\mathrm{P}^B \neq \mathrm{NP}^B$.

For parts 1, 2, and 3, by the proof of Baker et al. [1, Theorem 3] that $\mathrm{P}^B \neq \mathrm{NP}^B$, for every $k \in \mathbb{N}$ there exists at most one string in B of length k. Their proof also yields the language $L = \{0^{|y|} \mid y \in B\}$ such that L is in $\mathrm{NP}^B - \mathrm{P}^B$ (and L is also a member of $\mathrm{UP}^B - \mathrm{P}^B$). If $L \in \mathrm{SelfNP}^B$ or $L \in \mathrm{PermUP}^B$, then, by Lemma 4.8 (which relativizes), $L \in \mathrm{P}^B$, which is a contradiction. Thus, $\mathrm{SelfNP}^B \neq \mathrm{UP}^B$, $\mathrm{PermUP}^B \neq \mathrm{UP}^B$, and $\mathrm{SelfNP}^B \neq \mathrm{NP}^B$.

For part 4, let L' be defined as follows:

$$L' = \{x \mid x \in L \vee x \in B\}.$$

It is easy to see that this language is in $\mathrm{PermUP}^B - \mathrm{P}^B$, thus $\mathrm{P}^B \neq \mathrm{PermUP}^B$. □

In proving Theorem 4.4, we found it useful to employ a self-contained witnessing NPTM. We showed that all languages accepted by an honest, self-contained witnessing NPTM are in SelfNP. But what happens when we consider self-contained witnessing NPTMs that are not necessarily honest? Define $\mathrm{Self}_{\subseteq}\mathrm{NP}$ to be the class of all languages L for which there exists a self-contained witnessing NPTM that accepts L. Clearly, $\mathrm{SelfNP} \subseteq \mathrm{Self}_{\subseteq}\mathrm{NP}$. But is $\mathrm{SelfNP} = \mathrm{Self}_{\subseteq}\mathrm{NP}$? A language L is P-*immune* if it is of infinite cardinality and there is no subset of L that has infinite cardinality and is polynomial-time computable ([3], see also [2,20]).

Proposition 4.11. *If* $L \in \text{Self}_{\subseteq}\text{NP} - \text{SelfNP}$, *then* L *is* P-*immune*.

Proof. Suppose that $L \in \text{Self}_{\subseteq}\text{NP}$ via a self-contained witnessing NPTM M and that L is not P-immune. We will prove that $L \in \text{SelfNP}$. Suppose that the running time of M is bounded in the length of its inputs by a polynomial of degree $d \in \mathbb{N}$. Then there exists a set $L' \subseteq L$ of infinite cardinality that is in P. Choose $e \in \mathbb{N}$ such that $e > d$. The following NPTM M' accepts L.

> On input $x \in \Sigma^*$, if $x \notin L'$, then simulate M on input x (making exactly the same nondeterministic choices as M on input x would) and accept if and only if M accepts. Otherwise (i.e., in the case that $x \in L'$), nondeterministically guess a string w of length less than or equal to $\log^{\frac{1}{e}} |x|$. Now, *deterministically* simulate the computation of $M(w)$, that is, sequentially, in lexicographic order of the computation paths of $M(w)$, check, for each computation path z' in $M(w)$, whether $M(w)$ on z' accepts and if it does, accept. If each computation path in $M(x)$ rejects, reject.

We claim that $\bigcup_{x \in L} \text{wit}_{M'}(x) = L$. Choose $y \in \bigcup_{x \in L} \text{wit}_{M'}(x)$. Thus, for some $z \in L$, $y \in \text{wit}_{M'}(z)$. If $z \notin L'$, then by the definition of M', y was guessed during a simulation of M, in which case (since M is self-contained witnessing) $y \in L$. Otherwise, $z \in L'$, so by the definition of M', $|y| \leqslant \log^{\frac{1}{e}} |z|$ and $y \in L$. Thus, $\bigcup_{x \in L} \text{wit}_{M'}(x) \subseteq L$. Now, choose $y \in L$. For any z such that $z \in L'$ and $|y| \leqslant \log^{\frac{1}{e}} |z|$, by the definition of M', $y \in \text{wit}_{M'}(z)$ (since L' is of infinite cardinality, such a z always exists). Thus, $\bigcup_{x \in L} \text{wit}_{M'}(x) \supseteq L$. From this it follows that $\bigcup_{x \in L} \text{wit}_{M'}(x) = L$, and so, by the definition of SelfNP, $L \in \text{SelfNP}$. □

5. Conclusions and open questions

We showed that one-way permutations exist if and only if $\text{P} \neq \text{UP} \cap \text{coUP}$. Thus, the existence of one-way permutations is equivalent to a number of previously studied hypotheses [7,11,12,15,19,23].

We studied the self-witnessing language classes PermUP and SelfNP. We showed that the closure of PermUP under polynomial-time, one-to-one reductions is UP and that if PermUP = UP, then E = UE. We showed that $\text{SAT} \in \text{SelfNP}$ (thus, NP is the closure of SelfNP under polynomial-time, many-one reductions) and that, if SelfNP = NP, then E = NE. SelfNP can thus be viewed as a natural NP analog of PermUP.

Fig. 3 shows the known containment relations between the main classes studied in this paper.

Having developed a theory of self-witnessing languages, we hope it will be useful in studying additional open problems in complexity theory. For instance, part 4 of Corollary 4.7 shows that all languages reducible to SAT via a polynomial-time computable, honest, onto reduction are in SelfNP. Berman and Hartmanis famously conjectured [5] that all NP-complete languages are pairwise reducible to each other via a polynomial-time computable, polynomial-time invertible, onto, one-to-one reduction. This is known as the *Isomorphism Conjecture*. It could be the case that all NP-complete languages are self-witnessing, even if the Isomorphism Conjecture fails. This leads to the following conjecture.

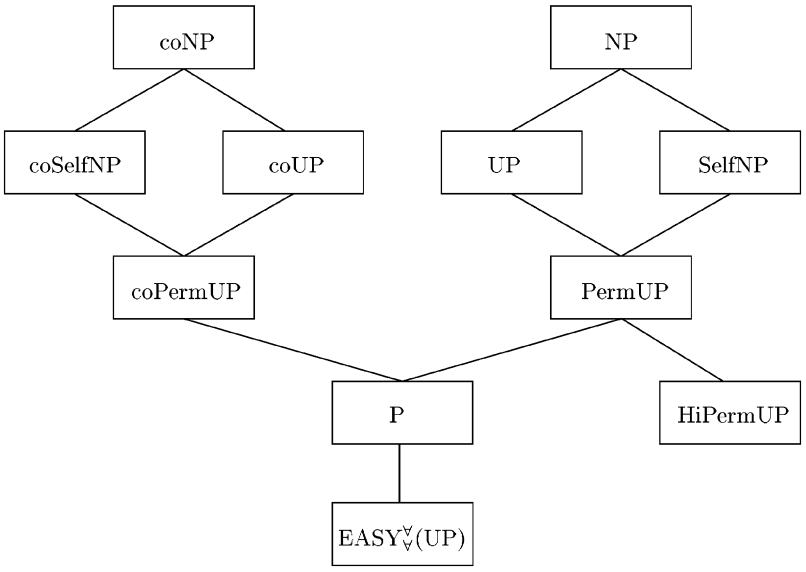

Fig. 3. The known containment relationships between the classes studied in this paper.

Conjecture 5.1. *All* NP-*complete languages are in* SelfNP.

Note that if Conjecture 5.1 does not hold, then, by part 4 of Corollary 4.7, the Isomorphism Conjecture does not hold. Conversely, we ask, "If Conjecture 5.1 holds, does the Isomorphism Conjecture necessarily hold?" As noted by Berman and Hartmanis [5], if the Isomorphism Conjecture holds, then $P \neq NP$. It follows that, if the answer to our question is "yes" and Conjecture 5.1 holds, then $P \neq NP$.

Another idea is to explore generalizations of SelfNP and PermUP. For instance, let SelfUP $= \{L \mid$ there exists a UPTM U such that $L(U) = L$ and $\mathrm{wit}_U(L) = L\}$. Does PermUP = SelfUP? What are the complexity-theoretic consequences of this equality holding?

Acknowledgments

We thank Lane Hemaspaandra for helpful comments and for allowing us to include here Theorem 4.9 and Lemma 4.8. We also thank Jörg Rothe for key insights, and Alina Beygelzimer and William Scherer for helpful discussions, and anonymous referees for helpful suggestions.

References

[1] T. Baker, J. Gill, R. Solovay, Relativizations of the P = ?NP question, SIAM J. Comput. 4 (4) (1975) 431–442.

[2] C. Bennett, J. Gill, Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1, SIAM J. Comput. 10 (1981) 96–113.

[3] L. Berman, On the structure of complete sets, in: Proceedings of the 17th IEEE Symposium on Foundations of Computer Science, IEEE Computer Science Press, Silver Spring, MD, 1976, pp. 76–80.
[4] L. Berman, Polynomial reducibilities and complete sets, Ph.D. Thesis, Cornell University, Ithaca, NY, 1977.
[5] L. Berman, J. Hartmanis, On isomorphisms and density of NP and other complete sets, SIAM J. Comput. 6 (2) (1977) 305–322.
[6] R. Book, Tally languages and complexity classes, Inform. Control 26 (2) (1974) 186–193.
[7] S. Fenner, L. Fortnow, A. Naik, J. Rogers, On inverting onto functions, in: Proceedings of the 11th Annual IEEE Conference on Computational Complexity, IEEE Computer Society Press, Silver Spring, MD, 1996, pp. 213–222.
[8] A. Goldberg, M. Sipser, Compression and ranking, SIAM J. Comput. 20 (3) (1991) 524–536.
[9] O. Goldreich, L. Levin, A hard-core predicate for any one-way function, in: Proceedings of the 21st ACM Symposium on Theory of Computing, ACM Press, New York, 1989, pp. 25–32.
[10] E. Grädel, Definability on finite structures and the existence of one-way functions, Methods Logic Comput. Sci. 1 (3) (1994) 299–314.
[11] J. Grollmann, A. Selman, Complexity measures for public-key cryptosystems, SIAM J. Comput. 17 (2) (1988) 309–335.
[12] J. Hartmanis, L. Hemachandra, Complexity classes without machines: on complete languages for UP, Theoret. Comput. Sci. 58 (1–3) (1988) 129–142.
[13] L. Hemaspaandra, Personal Communication, October 2000.
[14] L. Hemaspaandra, S. Jha, Defying upward and downward separation, Inform. Comput. 121 (1) (1995) 1–13.
[15] L. Hemaspaandra, J. Rothe, Characterizing the existence of one-way permutations, Theoret. Comput. Sci. 244 (1–2) (2000) 257–261.
[16] L. Hemaspaandra, J. Rothe, G. Wechsung, Easy sets and hard certificate schemes, Acta Inform. 34 (11) (1997) 859–879.
[17] C. Homan, Low ambiguity in strong, total, associative, one-way functions, Technical Report TR734, Computer Science Department, University of Rochester, August 2000.
[18] R. Impagliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in: Proceedings of the 21st ACM Symposium on Theory of Computing, ACM Press, New York, 1989, pp. 44–61.
[19] K. Ko, On some natural complete operators, Theoret. Comput. Sci. 37 (1) (1985) 1–30.
[20] K. Ko, D. Moore, Completeness, approximation, and density, SIAM J. Comput. 10 (4) (1981) 787–796.
[21] L. Levin, One-way functions and pseudorandom number generators, Combinatorica 7 (4) (1987) 357–363.
[22] R. Rao, J. Rothe, O. Watanabe, Upward separation for FewP and related classes, Inform. Process. Lett. 52 (4) (1994) 175–180 (corrigendum: Inform. Process. Lett. 74 (1–2) (2000) 89).
[23] J. Rothe, L. Hemaspaandra, On characterizing the existence of partial one-way permutations, Inform. Process. Lett. 82 (3) (2002) 165–171.
[24] S. Rudich, Limits on the provable consequences of one-way functions, Ph.D. Thesis, University of California, Berkeley, November 1988.
[25] A. Selman, A survey of one-way functions in complexity theory, Math. Systems Theory 25 (3) (1992) 203–221.
[26] L. Valiant, The relative complexity of checking and evaluating, Inform. Process. Lett. 5 (1) (1976) 20–23.
[27] O. Watanabe, On hardness of one-way functions, Inform. Process. Lett. 27 (3) (1988) 151–157.
[28] A. Yao, Theory and applications of trapdoor functions, in: Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Silver Spring, MD, 1982, pp. 80–91.