ONE-WAY PERMUTATIONS AND SELF-WITNESSING LANGUAGES

Christopher M. Homan[*]
Department of Computer Science
University of Rochester, Rochester NY 14627
choman@cs.rochester.edu

Mayur Thakur
Department of Computer Science
University of Rochester, Rochester NY 14627
thakur@cs.rochester.edu

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 also 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 nontrivial 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.

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

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. We prove that one-way permutations exist exactly if $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$. UP [Val76] 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 [GS88] and, independently, Ko [Ko85] (see also work by Berman [Ber77]) 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 [Ko85,GS88, HH88,Grä94,FFNR96,HR00,RH02], including the following.

1 The weak definability principle does not hold for some logic that is closed under first order operations [Grä94].

2 $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{UP}$ [RH02].

3 There exist UPTMs M and N such that $L(M) \subseteq L(N)$ and $f \notin \mathrm{FP}$, where f is any function having the property that, for all $x \in L(M)$, $f(\langle x, \mathrm{wit}_{\mathrm{M}}(x)\rangle) = \mathrm{wit}_{\mathrm{N}}(x)$.

$\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ [RH02] is the class of all languages in P for which there is an unambiguous Turing machine U accepting the language such that the mapping between a member of the language and its unique witness (i.e., the bits guessed by a nondeterministic accepting path of U) is not polynomial-time computable. Item 3 above follows by analogy to a result by Fenner et al. [FFNR96], 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 7, 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, PermUP = $\{L \mid$ there exists a UPTM U such that $L(U) = L$ and $\mathrm{wit}_{\mathrm{U}}$ permutes $L\}$, where $\mathrm{wit}_{\mathrm{U}}$ denotes a function that maps each $x \in L$ to its unique witness in U.

Clearly PermUP $\subseteq$ UP. Furthermore, it is easy to see that any language $L \in$ P is in PermUP via the following "simple UPTM" (i.e., a UPTM whose witness function $\mathrm{wit}_{\mathrm{U}}$ 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 $y = x$.

Based on the the above example, one might be tempted to regard PermUP as a characterization of simplicity. 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 either simple (i.e., accepted by some simple UPTM) or 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 HiPermUP = $\{L \mid$ there exists a UPTM U such that $L = L(U)$, $\mathrm{wit}_{\mathrm{U}}$ permutes L, and $\mathrm{wit}_{\mathrm{U}}$ is not polynomial-time computable$\}$, which is a (possibly empty) subset of PermUP. Intuitively, 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 $\cap$ 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 $\{L \mid$ there exists a nondeterministic Turing machine N that runs in polynomial time such that $L(N) = L$ and $\bigcup_{x \in L} \mathrm{wit}_{\mathrm{N}}(x) = L\}$, where $\mathrm{wit}_{\mathrm{N}}$ 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 SelfNP = NP, then E = NE). Furthermore, SAT $\in$ 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 is the conclusion. *(Note: Due to space limitations, some of the proofs have been omitted. All omitted proofs can be found in [HT01].)*

2. Definitions and Notations

All sets, 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 $S \subseteq \Sigma^*$, $wS = \{ws \mid s \in S\}$. 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." NP (respectively, NE) denotes the class of all languages L such that L is accepted by an NPTM (respectively, NETM). UPTM (respectively, UETM [RRW94, HJ95]) 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 [Val76] (respectively, UE [RRW94,HJ95]) is the class of all languages L such that L is accepted by some UPTM (respectively, UETM). TALLY is the class of all tally languages, that is, TALLY $= \{L \mid L \subseteq 1^*\}$. The set of total, deterministic, polynomial-time computable functions is called FP.

For each complexity class $\mathcal{C}$ and all a, b such that $\leq_b^a$ is defined, the *reduction closure* of $\mathcal{C}$ relative to $\leq_b^a$, denoted $\mathrm{R}_b^a(\mathcal{C})$, is the set $\{L \subseteq \Sigma^* \mid (\exists L' \in \mathcal{C})[L \leq_b^a L']\}$.

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 $x \in L(N)$ to the set of witnesses for x relative to N as $\mathrm{wit}_{\mathrm{N}} : \Sigma^* \rightarrow 2^{\Sigma^*}$ (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 $\mathrm{wit}_{\mathrm{N}}$ as a mapping from $L(N)$ to Σ^*.

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

For any set $S \subseteq \Sigma^*$ a function $f : \Sigma^* \rightarrow \Sigma^*$ 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 $f : \Sigma^* \rightarrow \Sigma^*$ is *polynomial-time-invertible* (alternatively, *FP-time invertible*) if there is a function $g \in$ 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 1 [Ber77,Ko85,GS88,Sel92] *A function $f : \Sigma^* \rightarrow \Sigma^*$ 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*

$|x| \leq p(|y|)$.

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 [GS88] provided the first independent study of complexity-theoretic, (one-to-one,) one-way functions (see also [Ber77,Ko85]). The definition below is for complexity-theoretic one-way functions of arbitrary ambiguity [Wat88].

Definition 2 [Wat88] (see [Ber77,Ko85]) *A function $f : \Sigma^* \to \Sigma^*$ is* one-way *if f is honest, polynomial-time computable, and not polynomial-time invertible.*

Definition 3 (see [Wat88,Hom00]) *For $g : \mathbb{N} \to \mathbb{N}$, we say a function $\sigma : \Sigma^* \to \Sigma^*$ is* g-to-1 *if*

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

3. One-Way Permutations

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

Theorem 4 P $\neq$ UP $\cap$ coUP *if and only if one-way permutations exist.*

As noted in the introduction, it is known (see [Ko85,GS88,Ber77]) that P $\neq$ UP $\cap$ 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 [Ko85, GS88,HH88,Grä94,FFNR96,HR00,RH02]). Theorem 5 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 [Ko85], of items 1–4 to Grollmann and Selman [GS88], of items 2 and 5 to Hartmanis and Hemachandra [HH88], of items 3 and 5–8 to Rothe and Hemaspaandra [RH02], and of 3 and 9 to Grädel [Grä94]. 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. [FFNR96].

Theorem 5 *The following are equivalent.*

1 *There exists a partial, one-to-one, onto, one-way function.*

2 P $\neq$ UP $\cap$ coUP.

3 *There exists a partial, one-to-one, one-way function f such that* $\mathrm{im}(f) \in$ P.

4 *There exists a total, one-to-one, one-way function f such that* $\mathrm{im}(f) \in$ P.

5 $\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \neq$ P.

6 $1\text{–}\mathrm{EASY}_{\forall}^{\forall}(\mathrm{UP}) \neq$ 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* UPTMs M *and* N *such that* $L(M) \subseteq L(N)$ *and* $f \notin \mathrm{FP}$, *where* f *is any function 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 [RH02] define $1\text{--}\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ as the set of all languages L in UP for which there exists a UPTM U that accepts L and 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. [FFNR96] consider the "NP version" of $1\text{--}\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$, however their results regarding this "NP version" [FFNR96] 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.

Theorem 6 [HR00] *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 4, we have the following.

Theorem 7 *The following are equivalent.*

1 *Any item from Theorem 5.*

2 *Any item from Theorem 6.*

3 $\Sigma^* \in \mathrm{HiPermUP}$.

4 *There exists* $L \in \mathrm{P}$ *such that* $L \in \mathrm{HiPermUP}$.

We now prove Theorem 4.

Proof of Theorem 4 The proof follows immediately from the equivalence of items 1 and 2 of Theorem 5 and the application of Lemma 8 for $g(n) = 1$.

Lemma 8 *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

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

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

The intuition behind the proof is captured by Figures 1 and 2. Figure 1 is

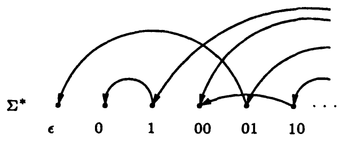

Figure 1. A graph where the vertices are the elements of Σ^*, and the edges are the mapping defined by h.

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 $f(x) = y$. 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 not necessarily total, each vertex of the graph in Figure 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 Figure 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 Figure 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

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

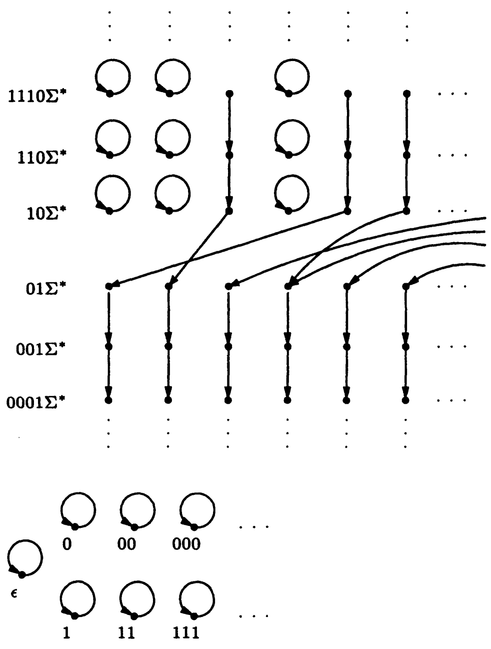

Figure 2. A graph where the vertices are the elements of Σ^*, and the edges are the mapping defined by f.

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, $||h^{-1}(z')|| \leq g(|z'|) \leq g(|z|)$. 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. ∎

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 9 $\mathrm{UP} = \mathrm{R}^p_{1\text{-}1}(\mathrm{PermUP})$.

As an immediate corollary to Theorem 9, we get the following.

Corollary 10 $\mathrm{P} \neq \mathrm{UP} \iff \mathrm{P} \neq \mathrm{PermUP}$.

The following theorem shows that $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$ if and only if $\mathrm{P} \neq \mathrm{PermUP} \cap \mathrm{coPermUP}$.

Theorem 11 $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP} \iff \mathrm{P} \neq \mathrm{PermUP} \cap \mathrm{coPermUP}$.

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 12 shows that SAT is a member of SelfNP.

Theorem 12 $\mathrm{SAT} \in \mathrm{SelfNP}$.

From this result, the corollary below easily follows.

Corollary 13 *1* $\mathrm{P} \neq \mathrm{NP} \iff \mathrm{P} \neq \mathrm{SelfNP}$.

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

3 $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \iff \mathrm{P} \neq \mathrm{SelfNP} \cap \mathrm{coSelfNP}$.

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

Corollaries 10 and 13, part 2 show that PermUP and SelfNP capture the hardest problems in UP and NP, respectively. Theorem 14, which follows from Lemma 15 below (both are due to Hemaspaandra [Hem00]), show that it is unlikely that either SelfNP = NP or PermUP = UP.

Theorem 14 *1* $\mathrm{PermUP} = \mathrm{UP} \Rightarrow \mathrm{E} = \mathrm{UE}$.

2 $\mathrm{SelfNP} = \mathrm{NP} \Rightarrow \mathrm{E} = \mathrm{NE}$.

Lemma 15 $\mathrm{TALLY} \cap \mathrm{SelfNP} \subseteq \mathrm{P}$.

We conclude with a relevant oracle result.

Theorem 16 *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 $\mathrm{P}^B \neq \mathrm{PermUP}^B$.

5. Conclusions and Open Questions

We showed that one-way permutations exist if and only if P $\neq$ UP $\cap$ coUP. Thus, the existence of one-way permutations is equivalent to a number of previously studied hypotheses [Ko85,GS88,HH88,FFNR96,RH02,HR00].

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 SAT $\in$ 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.

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

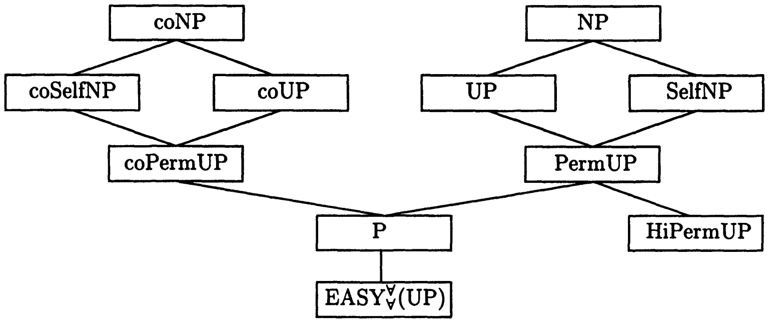

Figure 3. The known containment relationships between the classes studied in this paper.

Having developed a theory of self-witnessing languages, we hope it will be useful in studying additional open problems in complexity theory. For instance, Corollary 13, part 4 shows that all languages reducible to SAT via a polynomial-time computable, honest, onto reduction are in SelfNP. Berman and Hartmanis famously conjectured [BH77] 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.

Conjecture 17 *All* NP*-complete languages are in* SelfNP.

Note that if Conjecture 17 does not hold then, by Corollary 13, part 4, the Isomorphism Conjecture does not hold. Conversely, we ask, "If Conjecture 17 holds, does the Isomorphism Conjecture necessarily hold?" As noted by Berman and Hartmanis [BH77], if the Isomorphism Conjecture holds, then P $\neq$ NP. It follows that if the answer to our question is "yes" and Conjecture

17 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 $\mathrm{wit}_{\mathrm{U}}(L) = L\}$ and $\mathrm{Self}_{\subseteq}\mathrm{NP} = \{L \mid$ there exists an NPTM N such that $L(N) = L$ and $\bigcup_{x \in L} \mathrm{wit}_{\mathrm{N}}(x) \subseteq L\}$. Does PermUP = SelfUP? Does SelfNP = $\mathrm{Self}_{\subseteq}$NP? What are the complexity-theoretic consequences of either equality holding?

Acknowledgments

We thank Lane Hemaspaandra for helpful comments and for allowing us to include here Theorem 14 and Lemma 15. We also thank Jörg Rothe for key insights, Alina Beygelzimer and William Scherer for helpful discussions, and an anonymous referee for helpful suggestions and for pointing that in Corollary 13, number 4 we could change "polynomial-time invertible" to "honest" and still have a valid claim.

References

[Ber77] L. Berman. *Polynomial Reducibilities and Complete Sets.* PhD thesis, Cornell University, Ithaca, NY, 1977.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26(2):186–193, 1974.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 213–222. IEEE Computer Society Press, May 1996.

[Grä94] E. Grädel. Definability on finite structures and the existence of one-way functions. *Methods of Logic in Computer Science*, 1(3):299–314, 1994.

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

[Hem00] L. Hemaspaandra. Personal Communication, October 2000.

[HH88] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58(1-3):129–142, 1988.

[HJ95] L. Hemaspaandra and S. Jha. Defying upward and downward separation. *Information and Computation*, 121(1):1–13, 1995.

[Hom00] C Homan. Low ambiguity in strong, total, associative, one-way functions. Technical Report TR734, University of Rochester, Computer Science Department, August 2000. Thu, 10 Aug 00 13:36:44 GMT.

[HR00] L. Hemaspaandra and J. Rothe. Characterizing the existence of one-way permutations. *Theoretical Computer Science*, 244(1-2):257–261, 2000.

[HT01] C. Homan and M. Thakur. One-way permutations and self-witnessing languages. Technical Report 760, University of Rochester, 2001.

[Ko85] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37(1):1–30, 1985.

[RH02] J. Rothe and L. Hemaspaandra. On characterizing the existence of partial one-way permutations. *Information Processing Letters*, 82(3):165–171, 2002.

[RRW94] R. Rao, J. Rothe, and O. Watanabe. Upward separation for FewP and related classes. *Information Processing Letters*, 52(4):175–180, 1994. Corrigendum appears in same journal, Volume 74, number 1-2, page 89, 2000.

[Sel92] A. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25(3):203–221, 1992.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20–23, 1976.

[Wat88] O. Watanabe. On hardness of one-way functions. *Information Processing Letters*, 27(3):151–157, 1988.