

Closure and Nonclosure Properties of the Compressible and Rankable Sets

Jackson Abascal¹, Lane A. Hemaspaandra², Shir Maimon^{1,3}, and Daniel Rubery

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

November 5, 2016; revised October 30, 2018

Abstract

The rankable and compressible sets have been studied for more than a quarter of a century, ever since Allender [1] and Goldberg and Sipser [6] introduced the formal study of polynomial-time ranking. Yet even after all that time, whether the rankable and compressible sets are closed under the most important boolean and other operations remains essentially unexplored. The present paper studies these questions for both polynomial-time and recursion-theoretic compression and ranking, and for almost every case arrives at a Closed, a Not-Closed, or a Closed-Iff-Well-Known-Complexity-Classes-Collapse result for the given operation. Even though compression and ranking classes are capturing something quite natural about the structure of sets, it turns out that they are quite fragile with respect to closure properties, and many fail to possess even the most basic of closure properties. For example, we show that with respect to the join (aka disjoint union) operation: the P-rankable sets are not closed, whether the semistrongly P-rankable sets are closed is closely linked to whether $P = UP \cap \text{co}UP$, and the strongly P-rankable sets are closed.

2012 ACM Subject Classification Theory of computation \rightarrow Computational complexity and cryptography; Theory of computation \rightarrow Complexity classes; Theory of computation \rightarrow Computability; Information systems \rightarrow Data compression

Keywords and phrases complexity theory, closure properties, compression, ranking, computability

1 Introduction

Loosely put, a compression function f for a set A is a function over the domain Σ^* such that (a) $f(A) = \Sigma^*$ and (b) $(\forall a, b \in A : a \neq b)[f(a) \neq f(b)]$. That is, f puts A in 1-to-1 correspondence with Σ^* . This is sometimes described as providing a minimal perfect hash function for A : It is perfect since there are no collisions (among elements of A), and it is minimal since not a single element of the codomain is missed. Note that the above does not put any constraints on what strings the elements of \overline{A} are mapped to, or even about whether the compression function needs to be defined on such strings. A ranking function is similar, yet stronger, in that a ranking function sends the i th string in A to the integer i ; it respects the ordering of the members of A .

The study of ranking was started by Allender [1] and Goldberg and Sipser [6], and has been pursued in many papers since, especially in the early 1990s, e.g., [9, 14, 5, 2]. The study of ranking led to the study of compression, which was started—in its current form,

¹ Supported in part by a CRA-W Collaborative Research Experiences for Undergraduates (CREU) grant.

² This work was done in part while on a sabbatical stay at ETH Zürich and the University of Düsseldorf.

³ Current affiliation: Department of Computer Science, Cornell University

though already foreshadowed in a notion of [6]—by Goldsmith, Hemachandra, and Kunen [7] (see also [8]). The abovementioned work focused on polynomial-time or logarithmic-space ranking or compression functions. More recently, both compression and ranking have also been studied in the recursion-theoretic context ([11], and see the discussion therein for precursors in classic recursive function theory), in particular for both the case of (total) recursive compression/ranking functions (which of course must be defined on all inputs in Σ^*) and the case of partial-recursive compression/ranking functions (i.e., functions that on some or all elements of the complement of the set being compressed/ranked are allowed to be undefined).

In the present paper, we continue the study of both complexity-theoretic and recursion-theoretic compression and ranking functions. In particular, the earlier papers often viewed the compressible sets or the rankable sets as a *class*. We take that very much to heart, and seek to learn whether these classes do, or do not, possess key closure properties. Our main contributions can be seen in Table 1, where we obtain closure and nonclosure results for many previously studied variations of compressible and rankable sets under boolean operations (Section 4). We also study the closure of these sets under additional operations, such as the join, aka disjoint union (Section 5). And we introduce the notion of compression *onto a set* and characterize the robustness of compression under this notion. In particular, by a finite-injury priority argument with some interesting features we show that there exist RE sets that each compress to the other, yet that nonetheless are not recursively isomorphic (Section 3).

2 Definitions

Throughout this paper, “P” when used in a function context (e.g., the P-rankable sets) will denote the class of total, polynomial-time computable functions from Σ^* to Σ^* . Additionally, throughout this paper, $\Sigma = \{0, 1\}$. FREC will denote the class of total, recursive functions

Class	\cap	\cup	complement
strong-P-rankable	$P = P^{\#P}$ (Th. 4.2)	$P = P^{\#P}$ (Th. 4.2)	Yes (Prop. 4.3)
semistrong-P-rankable	$P = P^{\#P}$ (Th. 4.2)	$P = P^{\#P}$ (Th. 4.2)	$\approx P = UP \cap \text{co}UP$ (Th. 4.6, Cor 4.9)
P-rankable, P-compressible', FREC -rankable, FREC -com- pressible, FPR -rankable, and FPR -compressible	No (Th. 4.10)	No (Th. 4.11)	No (Th. 4.12)
strong-P-rankable ^c	No (Th. 4.13)	No (Th. 4.13)	Yes (Prop. 4.3)
semistrong-P-rankable ^c	No (Th. 4.13)	No (Th. 4.13)	$\approx P = UP \cap \text{co}UP$ (Th. 4.6, Cor 4.9)
P-rankable ^c , P-compressible ^c , FREC -rankable ^c , FREC -com- pressible ^c , FPR -rankable ^c , and FPR -compressible ^c	No (Th. 4.13)	No (Th. 4.13)	No (Th. 4.12)

Table 1 Overview of results for closure of these classes under boolean operations. If an entry does not contain “No” or “Yes” then the class is closed under the operation if and only if the entry holds. A special case is semistrong-P-rankable and semistrong-P-rankable^c, in which we deliberately use the \approx symbol to indicate that the implication is true in one direction and in the other direction currently is known to be true only for a broad subclass of these sets. Specifically, if $P = UP \cap \text{co}UP$ then the complements of all “nongappy” semistrong-P-rankable sets are themselves semistrong-P-rankable.

from Σ^* to Σ^* . F_{PR} will denote the class of partial recursive functions from Σ^* to Σ^* . ϵ will denote the empty string. We define the function $\text{shift}(x, n)$ for $n \in \mathbb{Z}$. If $n \geq 0$, then $\text{shift}(x, n)$ is the string n spots after x in lexicographical order, e.g., $\text{shift}(\epsilon, 4) = 01$. For $n > 0$, define $\text{shift}(x, -n)$ as the string n spots before x in lexicographical order, or ϵ if no such string exists. We define the symmetric difference $A \triangle B = (A - B) \cup (B - A)$. The symbol \mathbb{N} will denote the natural numbers $\{0, 1, 2, 3, \dots\}$.

We now define the notions of compressible and rankable sets.

► **Definition 2.1** (Compressible sets [11]).

1. Given a set $A \subseteq \Sigma^*$, a (possibly partial) function f is a *compression function* for A exactly if
 - a. $\text{domain}(f) \supseteq A$,
 - b. $f(A) = \Sigma^*$, and
 - c. for all a and b in A , if $a \neq b$ then $f(a) \neq f(b)$.
2. Let \mathcal{F} be any class of (possibly partial) functions mapping from Σ^* to Σ^* . A set A is \mathcal{F} -*compressible* if some $f \in \mathcal{F}$ is a compression function for A .
3. For each \mathcal{F} as above, \mathcal{F} -compressible = $\{A \mid A \text{ is } \mathcal{F}\text{-compressible}\}$ and \mathcal{F} -compressible' = \mathcal{F} -compressible $\cup \{A \subseteq \Sigma^* \mid A \text{ is a finite set}\}$.
4. For each \mathcal{F} as above and each $\mathcal{C} \subseteq 2^{\Sigma^*}$, we say that \mathcal{C} is \mathcal{F} -*compressible* if all infinite sets in \mathcal{C} are \mathcal{F} -compressible.

Note that a compression function f for A can have any behavior on elements of \bar{A} and need not even be defined. Finite sets cannot have compression functions as they do not have enough elements to be mapped onto Σ^* . Thus part 4 of Definition 2 defines a class to be \mathcal{F} -compressible if and only if its *infinite* sets are \mathcal{F} -compressible.

Ranking can be informally thought of as a sibling of compression that preserves lexicographical order within the set. We consider three classes of rankable functions that differ in how they are allowed to behave on the complement of the set they rank. Although ever since the paper of Hemachandra and Rudich [9], which introduced two of the three types, there have been those three types of ranking classes, different papers have used different (and sometimes conflicting) terminology for these types. Here, we use the (without modifying adjective) terms “ranking function” and “rankable” in the same way as Hemaspaandra and Rubery [11] do, for the least restrictive form of ranking (the one that can even “lie” on the complement). That is the form of ranking that is most naturally analogous with compression, and so it is natural that both terms should lack a modifying adjective. For the most restrictive form of ranking, which even for strings x in the complement of the set A being ranked must determine the number of strings up to x that are in A , like Hemachandra and Rudich [9] we use the terms “strong ranking function” and “strong(ly) rankable.” And for the version of ranking that falls between those two, since for strings in the complement it need only detect that they are in the complement, we use the terms “semistrong ranking function” and “semistrong(ly) rankable.”

► **Definition 2.2** ([1, 6]). $\text{rank}_A(y) = \|\{z \mid z \leq y \wedge z \in A\}\|$.

► **Definition 2.3** (Rankable sets, [1, 6], see also [11]).

1. Given a set $A \subseteq \Sigma^*$, a (possibly partial) function f is a *ranking function* for A exactly if
 - a. $\text{domain}(f) \supseteq A$ and
 - b. if $x \in A$, then $f(x) = \text{rank}_A(x)$.
2. Let \mathcal{F} be any class of (possibly partial) functions mapping from Σ^* to Σ^* . A set A is \mathcal{F} -*rankable* if some $f \in \mathcal{F}$ is a ranking function for A .

3. For each \mathcal{F} as above, \mathcal{F} -rankable = $\{A \mid A \text{ is } \mathcal{F}\text{-rankable}\}$.
4. For each \mathcal{F} as above and each $\mathcal{C} \subseteq 2^{\Sigma^*}$, \mathcal{C} is \mathcal{F} -rankable if all sets in \mathcal{C} are \mathcal{F} -rankable.

► **Definition 2.4** (Semistrongly rankable sets, [9], see also [11]).

1. Given a set $A \subseteq \Sigma^*$, a function f is a *semistrong ranking function* for A exactly if
 - a. $\text{domain}(f) = \Sigma^*$,
 - b. if $x \in A$, then $f(x) = \text{rank}_A(x)$, and
 - c. if $x \notin A$, $f(x)$ indicates “not in set” (e.g., via the machine computing f halting in a special state; we still view this as a case where x belongs to $\text{domain}(f)$).
2. Let \mathcal{F} be any class of functions mapping from Σ^* to Σ^* . A set A is *semistrong- \mathcal{F} -rankable* if some $f \in \mathcal{F}$ is a semistrong ranking function for A .
3. For each \mathcal{F} as above, $\text{semistrong-}\mathcal{F}\text{-rankable} = \{A \mid A \text{ is semistrong-}\mathcal{F}\text{-rankable}\}$.
4. For each \mathcal{F} as above and each $\mathcal{C} \subseteq 2^{\Sigma^*}$, we say that \mathcal{C} is *semistrong- \mathcal{F} -rankable* if all sets in \mathcal{C} are semistrong- \mathcal{F} -rankable.

► **Definition 2.5** (Strongly rankable sets, [9], see also [11]).

1. Given a set $A \subseteq \Sigma^*$, a function f is a *strong ranking function* for A exactly if
 - a. $\text{domain}(f) = \Sigma^*$ and
 - b. $f(x) = \text{rank}_A(x)$.
2. Let \mathcal{F} be any class of functions mapping from Σ^* to Σ^* . A set A is *strong- \mathcal{F} -rankable* exactly if $(\exists f \in \mathcal{F})[f \text{ is a strong ranking function for } A]$.
3. For each \mathcal{F} as above, $\text{strong-}\mathcal{F}\text{-rankable} = \{A \mid A \text{ is strong-}\mathcal{F}\text{-rankable}\}$.
4. For each \mathcal{F} as above and each $\mathcal{C} \subseteq 2^{\Sigma^*}$, we say that \mathcal{C} is *strong- \mathcal{F} -rankable* if all sets in \mathcal{C} are strong- \mathcal{F} -rankable.

For almost any natural class of functions, \mathcal{F} , we will have that \mathcal{F} -rankable is contained in \mathcal{F} -compressible'. In particular, P , F_{PR} , and F_{REC} each have this property. If f is a ranking function for A (in the sense of part 1 of Definition 2.3), for our same-class compression function for A we can map $x \in \Sigma^*$ to the $f(x)$ -th string in Σ^* (where we consider ϵ to be the first string in Σ^*) if $f(x) > 0$, and if $f(x) = 0$ what we map to is irrelevant so map to any particular fixed string (for concreteness, ϵ).

For each class $\mathcal{C} \subseteq 2^{\Sigma^*}$, \mathcal{C}^c will denote the complement of \mathcal{C} , i.e., $2^{\Sigma^*} - \mathcal{C}$. For example, $P\text{-rankable}^c$ is the class of non- P -rankable sets.

The class semistrong- P -rankable is a subset of P (indeed, a strict subset unless $P = P^{\#P}$ [9]), but there exist undecidable sets that are P -rankable. Clearly, the class of semistrong- REC -rankable sets equals the class of strong- REC -rankable sets.

3 Compression onto B : Robustness with Respect to Target Set

A compression function for a set A is 1-to-1 and onto Σ^* when the function's domain is restricted to A . It is natural to wonder what changes when we switch target sets from Σ^* to some other set $B \subseteq \Sigma^*$. We now define this notion. In our definition, we do allow strings in \overline{A} to be mapped to B or to \overline{B} , or even, for the case of F_{PR} maps, to be undefined. In particular, this definition does not require that $f(\Sigma^*) = B$. Recall from Section 1 that, throughout this paper, $\Sigma = \{0, 1\}$.

► **Definition 3.1** (Compressible to B).

1. Given sets $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^*$, a (possibly partial) function f is a *compression function for A to B* exactly if
 - a. $\text{domain}(f) \supseteq A$,

- b. $f(A) = B$, and
 - c. for all a and b in A , if $a \neq b$ then $f(a) \neq f(b)$.
2. Let \mathcal{F} be any class of (possibly partial) functions mapping from Σ^* to Σ^* . A set A is \mathcal{F} -compressible to B if some $f \in \mathcal{F}$ is a compression function for A to B .

The classes \mathcal{F} of interest to us will be F_{REC} and F_{PR} . Clearly, compression is simply the $B = \Sigma^*$ case of this definition, e.g., a function f is a compression function for A if and only if f is a compression function for A to Σ^* , and set A is \mathcal{F} -compressible if and only if A is \mathcal{F} -compressible to Σ^* .

A natural first question to ask is whether compression to B is a new notion, or whether it coincides with our existing notion of compression to Σ^* , at least for sets B from common classes such as REC and RE . The following result shows that for REC and RE this new notion does coincide with our existing one.

► **Theorem 3.2.** *Let A and B be infinite sets.*

- 1. *If $B \in \text{REC}$, then A is F_{REC} -compressible to B if and only if A is F_{REC} -compressible to Σ^* .*
- 2. *If $B \in \text{RE}$, then A is F_{PR} -compressible to B if and only if A is F_{PR} -compressible to Σ^* .*

Proof. We first prove part 1, beginning with the “if” direction.

Suppose A is F_{REC} -compressible to Σ^* by a recursive function f , and suppose B is recursive and infinite. Let $f'(x)$ output the element $y \in B$ such that $\text{rank}_B(y) = f(x)$. Then f' is recursive, and A is F_{REC} -compressible to B by f' .

For the “only if” direction, let B be an infinite recursive set. Suppose that A is F_{REC} -compressible to B by a recursive function f . Let $f'(x) = \epsilon$ if $f(x)$ is not in B . Otherwise, let $f'(x) = \text{rank}_B(f(x))$. Then f' is recursive, and A is F_{REC} -compressible to Σ^* by f' .

Let us turn to part 2 of the theorem. Again, we begin with the “if” direction. Let B be an infinite RE set, and let E enumerate the elements of B without repetitions. Suppose A is F_{PR} -compressible to Σ^* by a partial recursive function f . Then f' does the following on input x .

- 1. Simulate $f(x)$. This may run forever if $x \notin \text{domain}(f)$.
- 2. If $f(x)$ outputs a value, simulate E until it enumerates $f(x)$ strings.
- 3. Output the $f(x)$ -th string enumerated by E .

The function f' is partial recursive, and A is F_{PR} -compressible to B via f' .

For the “only if” direction, let B be infinite and RE and let E be an enumerator for B . Suppose A is F_{PR} -compressible to B via a partial recursive function f . On input x , our f' will work as follows.

- 1. Simulate $f(x)$.
- 2. If $f(x)$ outputs a value, run E until it enumerates $f(x)$. This step may run forever if $f(x) \notin B$.
- 3. Suppose $f(x)$ is the l th string output by E . Then output the l th string in Σ^* .

f' is partial recursive, and A is F_{PR} -compressible to Σ^* by f' . ◀

Theorem 3.2 covers the two most natural pairings of set classes with function classes: recursive sets B with F_{REC} compression, and RE sets B with F_{PR} compression. What about pairing recursive sets under F_{PR} compression, or RE sets under recursive compression? We note as the following theorem that one and a half of the analogous statements hold, but the remaining direction fails.

- **Theorem 3.3.** 1. Let A and B be infinite sets and suppose that $B \in \text{REC}$. Then A is FPR -compressible to B if and only if A is FPR -compressible to Σ^* .
2. Let A and B be infinite sets with $B \in \text{RE}$. If A is FPR -compressible to Σ^* , then A is FPR -compressible to B . In fact, we may even require that the compression function for A to B satisfies $f(\Sigma^*) = B$.
3. There are infinite sets A and B with $B \in \text{RE}$ such that A is FREC -compressible to B but A is not FREC -compressible to Σ^* .

Proof. The first part follows immediately from Theorem 3.2, part 2. The second part follows as a corollary to the proof of Theorem 3.2, part 2. In particular, the proof of the “ \Leftarrow ” direction proves the second part, since it is clear that if f is a recursive function the f' defined there is also recursive.

The third part follows from [11] in which it is shown that any set in $\text{RE} - \text{REC}$ is not FREC -compressible to Σ^* . Thus if we let $A = B$ be any set in $\text{RE} - \text{REC}$, then A is FREC -compressible to B by the function $f(x) = x$ but B is not FREC -compressible to Σ^* . ◀

Another interesting question is how recursive compressibility to B is, or is not, linked to recursive isomorphism. Recall two sets A and B are recursively isomorphic if there exists a recursive bijection $f : \Sigma^* \rightarrow \Sigma^*$ with $f(A) = B$. Although recursive isomorphism of sets implies mutual compressibility to each other, we prove via a finite-injury priority argument that the converse does not hold (even when restricted to the RE sets). The argument has an interesting graph-theoretic flavor, and involves queuing infinitely many strings to be added to a set at once.

► **Theorem 3.4.** If $A \equiv_{\text{iso}} B$, then A is FREC -compressible to B and B is FREC -compressible to A .

► **Theorem 3.5.** There exist RE sets A and B such that A is FREC -compressible to B and B is FREC -compressible to A , yet $A \not\equiv_{\text{iso}} B$.

Proof of Theorem 3.4. Now A is FREC -compressible to B by simply letting our FREC -compression function be the recursive isomorphism function f . Since each recursive isomorphism has a recursive inverse, B is FREC -compressible to A by letting our FREC -compression function be the inverse of f . ◀

Proof of Theorem 3.5. Before defining A and B , we will define a function f which will serve as both a compression function from A to B and a compression function from B to A . First, fix a recursive isomorphism between Σ^* and $\{\langle t, j, k \rangle \mid t \in \{0, 1, 2, 3\} \wedge j, k \in \mathbb{N}\}$. Now we will define f as follows. For each $j, k \in \mathbb{N}$, let $f(\langle 3, j, k \rangle) = \langle 3, j + 1, k \rangle$. For each $j, k \in \mathbb{N}$, $j > 0$, and $t \in \{0, 1, 2\}$, let $f(\langle t, j, k \rangle) = \langle t, j - 1, k \rangle$. Finally, for each $k \in \mathbb{N}$, let $f(\langle 0, 0, k \rangle) = \langle 3, 0, k \rangle$, $f(\langle 1, 0, k \rangle) = \langle 0, 0, k \rangle$, and $f(\langle 2, 0, k \rangle) = \langle 3, 0, k \rangle$. Let $\ell : \Sigma^* \rightarrow \{0, 1\}$ be the unique function such that $\ell(\langle 0, 0, k \rangle) = 0$ for all $k \in \mathbb{N}$ and $\ell(f(x)) = 1 - \ell(x)$. Let D_f be the directed graph with edges $(x, f(x))$. Note that ℓ is a 2-coloring of D_f if we treat the edges as being undirected. See Figure 1.

Call a set C a *path set* if for all $x \in C$, $f(x) \in C$ and there is exactly one $y \in C$ such that $f(y) = x$. Suppose C is a path set. Let $C_i = \{x \in C \mid \ell(x) = i\}$ for $i \in \{0, 1\}$. By the assumed property of C , we have C_0 and C_1 are FREC -compressible to each other by f . Furthermore, if C is RE then so are C_0 and C_1 since $C_i = C \cap \{x \mid \ell(x) = i\}$ is the intersection of an RE set with a recursive set. If we provide an enumerator for a path set C such that $C_0 \not\equiv_{\text{iso}} C_1$, we may let $A = C_0$ and $B = C_1$ and be done.

Our enumerator for C proceeds in two interleaved types of stages: printing stages P_i and evaluation stages E_i . More formally, we proceed in stages labeled E_i and P_i for $i \geq 1$, interleaved as $E_1, P_1, E_2, P_2, \dots, E_n, P_n, \dots$ when running. We also maintain a set Q of elements of the form $\langle t, k \rangle$, where $t \in \{0, 1, 2\}$ and $k \in \mathbb{N}$. This set Q will only ever be added to as the procedure runs.

In the printing stage P_i , we do the following for every $\langle t, k \rangle$ in Q . Enumerate $\langle 3, j, k \rangle$ and $\langle t, j, k \rangle$ for all $j \leq i$. If $t = 1$, additionally enumerate $\langle 0, 0, k \rangle$. Adding an element $\langle t, k \rangle$ to Q in some evaluation stage E_i is essentially adding an infinite path of nodes in D_f to C .

In addition to Q , we also maintain an integer b and a set R of elements $\langle n, k \rangle$ where $n, k \in \mathbb{N}$. If $\langle n, k \rangle \in R$ after stage i , it signifies that we have not yet satisfied the condition that φ_n , the n th partial recursive function, is not an isomorphism function between C_0 and C_1 . In stage E_i we perform the following. Add $\langle i, b \rangle$ to R . Increment b by one. For each $\langle n, k \rangle \in R$, run φ_n , the n th partial recursive function, on $\langle 0, 0, k \rangle$ for i steps. If none of these machines halt in their allotted time, end the stage. Otherwise, let n_i be the smallest number such that φ_{n_i} produced an output $w_i = \langle x_i, y_i, z_i \rangle$ on its respective input $\langle 0, 0, k_i \rangle$. We now break into cases:

1. If $\ell(w_i) = 0$ add $\langle 0, k_i \rangle$ to Q .
2. If $z_i \neq k_i$ and $\ell(w_i) = 1$ and as it stands w_i would not be printed eventually if there were only type P stages from now on, add $\langle 0, k_i \rangle$ to Q .
3. If $z_i \neq k_i$ and $\ell(w_i) = 1$ and as it stands w_i would be printed eventually if there were only type P stages from now on, do nothing.
4. If $z_i = k_i$ and $\ell(w_i) = 1$ and $x_i = 0$, add $\langle 1, k_i \rangle$ to Q .
5. If $z_i = k_i$ and $\ell(w_i) = 1$ and either $x_i = 1$ or $x_i = 2$, add $\langle 0, k_i \rangle$ to Q .
6. If $z_i = k_i$ and $\ell(w_i) = 1$ and $x_i = 3$, add $\langle 2, k_i \rangle$ to Q .

Set $b = \max(k_i, z_i) + 1$. Remove all pairs $\langle n, k \rangle$ with $n \geq n_i$ from R . Then for each n from $n_i + 1$ to i , first add $\langle n, b \rangle$ and subsequently increment b by 1.

We will first prove that C is a path set. If $x \in C$, then it is printed in some printing stage P_i . By tracing the definition of f and the procedure for printing stages, one can verify that both $f(x)$ and exactly one y such that $f(y) = x$ will be printed in stage P_j for $j \geq i$. This string y will be the only one ever printed, since no two elements with the same second coordinate will ever be added to Q , as every element added to Q has the current state of b as its second coordinate, and b only ever strictly increases between additions to Q .

Let F_n be the condition that φ_n fails to be a recursive isomorphism of C_0 onto C_1 . Fix n . Say during E_i we have $n_i = n$. In cases 1, 2, 4, and 5, we force φ_n to map $\langle 0, 0, k_i \rangle \in C_0$ to something out of C_1 . In cases 3 and 6, we force φ_n to map $\langle 0, 0, k_i \rangle \notin C_0$ to something in C_1 . Thus whenever at stage i we have $n_i = n$, condition F_n becomes satisfied, though perhaps not permanently. Specifically, in case 2, w could be printed later to satisfy some other F_m and in doing so “injure” F_n . However, note that during E_i the variable b is set to $\max(k_i, z_i)$, thus F_n can only be injured when satisfying conditions F_m for $m < n$. Pairs

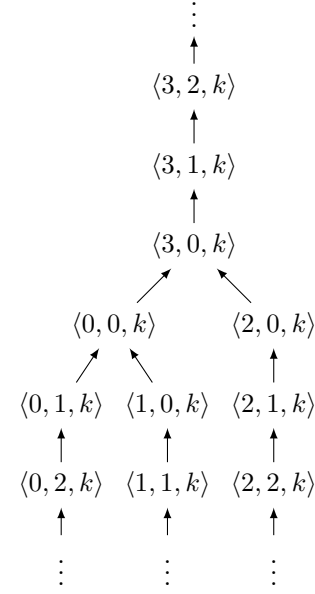


Figure 1 A diagram of D_f , for fixed k .

with first coordinate n will only ever be added to R when after satisfying some such F_m , in addition to once initially, so in total only a finite number of times. If φ_n always halts, F_n will eventually be satisfied and never injured again.

This proves that C is a path set such that $C_0 \not\equiv_{iso} C_1$. Thus C_0 and C_1 are RE sets that are F_{REC} -compressible to each other by f , but are not recursively isomorphic. \blacktriangleleft

For those interested in the issue of isomorphism in the context of complexity-theoretic functions, which was not the focus above, we mention that: Hemaspaandra, Zaki, and Zimand [13] prove that the P-rankable sets are not closed under \equiv_{iso}^P ; Goldsmith and Homer [8] prove that the strong-P-rankable sets are closed under \equiv_{iso}^P if and only if $P = P^{\#P}$; and [13] notes that the semistrong-P-rankable sets similarly are closed under \equiv_{iso}^P if and only if $P = P^{\#P}$.

4 Closures and Nonclosures under Boolean Operations

We now move on to a main focus of this paper, the closure properties of the compressible and the rankable sets. We explore these properties both in the complexity-theoretic and the recursion-theoretic domains. Table 1 on page 2 summarizes our findings.

► **Lemma 4.1.** *Let A and B be strong-P-rankable. Then $A \cup B$ is strong-P-rankable if and only if $A \cap B$ is.*

Proof. The identity $\text{rank}_{A \cap B}(x) + \text{rank}_{A \cup B}(x) = \text{rank}_A(x) + \text{rank}_B(x)$ allows us to compute either of $\text{rank}_{A \cap B}(x)$ or $\text{rank}_{A \cup B}(x)$ from the other. \blacktriangleleft

► **Theorem 4.2.** *The following conditions are equivalent:*

1. *the classes strong-P-rankable and semistrong-P-rankable are closed under intersection,*
2. *the classes strong-P-rankable and semistrong-P-rankable are closed under union, and*
3. $P = P^{\#P}$.

Proof. It was proven in [9] by Hemachandra and Rudich that $P = P^{\#P}$ implies $P = \text{strong-P-rankable} = \text{semistrong-P-rankable}$. Since P is closed under intersection and union, this shows that 3 implies 1 and 2. To show, in light of Lemma 4.1, that either 1 or 2 would imply 3, we will construct two strong-P-rankable sets whose intersection is not P-rankable unless $P = P^{\#P}$.

Let A_1 be the set of $x1y1$ such that $|x| = |y|$, x encodes a boolean formula, and y (padded with 0s so that it has length $|x|$) encodes a satisfying assignment for the formula x . Let A_0 be the set of $x1y0$ such that $|x| = |y|$, and $x1y1 \notin A_1$. Let A_2 be the set of strings $x0^{|x|+1}1$. Let $A = A_0 \cup A_1 \cup A_2$. For every x , and every y such that $|x| = |y|$, exactly one of $x1y0$ and $x1y1$ is in A . Thus, for any x , we can find $\text{rank}_{A_0 \cup A_1}(x)$ in polynomial time. Clearly A_2 is strong-P-rankable. Since $A_0 \cup A_1$ and A_2 are disjoint, $\text{rank}_{A_0 \cup A_1 \cup A_2}(x) = \text{rank}_{A_0 \cup A_1}(x) + \text{rank}_{A_2}(x)$, so A is strong-P-rankable.

Let $B = \Sigma^*1$. Then $A \cap B = A_1 \cup A_2$ is the set of $x1y1$ such that y encodes a satisfying assignment for x , along with all strings $x0^{|x|+1}1$. If $A_1 \cup A_2$ were P-rankable, then we could count satisfying assignments of a formula x in polynomial time by computing $\text{rank}_{A \cap B}(\text{shift}(x, 1)0^{|\text{shift}(x, 1)|+1}1) - \text{rank}_{A \cap B}(x0^{|x|+1}1) - 1$. Thus $\#SAT$ is polynomial-time computable and so $P = P^{\#P}$. \blacktriangleleft

► **Proposition 4.3.** *strong-P-rankable is closed under complementation.*

Proof. The identity $\text{rank}_A(x) + \text{rank}_{\overline{A}}(x) = \text{rank}_{\Sigma^*}(x)$ allows us to compute either of $\text{rank}_A(x)$ or $\text{rank}_{\overline{A}}(x)$ from the other. \blacktriangleleft

► **Corollary 4.4.** *The class strong-P-rankable^G is also closed under complementation.*

► **Lemma 4.5.** *The class semistrong-P-rankable is closed under complementation if and only if semistrong-P-rankable = strong-P-rankable.*

Proof. The “if” direction follows directly from Proposition 4.3. For the “only if” direction, let A be a semistrong-P-rankable set with ranking function r_A , and suppose \bar{A} is semistrong-P-rankable with semistrong ranking function $r_{\bar{A}}$. Then $\text{rank}_A(x) = r_A(x)$ if $x \in A$, and equals $\text{rank}_{\Sigma^*}(x) - r_{\bar{A}}(x)$ otherwise. The function r_A decides membership in A , so we can compute $\text{rank}_A(x)$ in polynomial time. ◀

► **Theorem 4.6.** *If semistrong-P-rankable is closed under complementation, then $P = \text{UP} \cap \text{coUP}$.*

Proof. Suppose semistrong-P-rankable is closed under complementation. Let A be in $\text{UP} \cap \text{coUP}$. Then there exists a UP machine U recognizing A , and a UP machine \hat{U} recognizing \bar{A} . If $x \in A$, let $f(x)$ be the unique accepting path for x in U . Otherwise, let $f(x)$ be the unique accepting path for x in \hat{U} . Choose a polynomial p such that, without loss of generality, $p(x)$ is monotonically increasing and $|f(x)| = p(|x|)$ (we may pad accepting paths with 0s to make this true).

The language $B = \{xf(x)1 \mid x \in \Sigma^*\} \cup \{x0^{p(|x|)+1} \mid x \in \Sigma^*\}$ is semistrong-P-rankable since $\text{rank}_B(x0^{p(|x|)+1}) = 2\text{rank}_{\Sigma^*}(x) - 1$ and $\text{rank}_B(xf(x)1) = 2\text{rank}_{\Sigma^*}(x)$. Since semistrong-P-rankable is closed under complementation, and B is semistrong-P-rankable, B is also strong-P-rankable by Lemma 4.5. Let x be a string, and let $y = \text{shift}(x, 1)$. We can binary search on the value of rank_B in the range from $x0^{p(|x|)+1}$ to $y0^{p(|y|)+1}$ to find the first value xz where $|z| = p(|x|) + 1$ and $\text{rank}_B(xz) = 2\text{rank}_{\Sigma^*}(x)$. See that $f(x)$ must equal z . We then simulate U on the path z and \hat{U} on the path z . Now z must be an accepting path for one of these machines, so either U accepts and $x \in A$, or \hat{U} accepts and $x \notin A$. ◀

► **Definition 4.7.** A set is *nongappy* if there exists a polynomial p such that, for each $n \in \mathbb{N}$, there is some element $y \in A$ such that $n \leq |y| \leq p(n)$.

► **Theorem 4.8.** *If $P = \text{UP} \cap \text{coUP}$ then each nongappy semistrong-P-rankable set is strong-P-rankable.*

Proof. Let A be a nongappy semistrong-P-rankable set, and let p be a polynomial such that, for each $n \in \mathbb{N}$, there is y in A such that $n \leq |y| \leq p(y)$. Let r be a polynomial-time semistrong ranking function for A . The coming string comparisons of course will be lexicographical. Let L be the set of $\langle x, b \rangle$ such that there exists at least one string in A that is less than or equal to x and b a prefix of the greatest string in A that is lexicographically less than or equal to x . L is in $\text{UP} \cap \text{coUP}$ by the following procedure. Let x_0 be the lexicographically first string in A . If $x < x_0$ output 0. Otherwise, guess a string $z > x$ such that $|z| \leq p(|x| + 1)$. Then guess a $y \leq x$. If y and z are in A and $r(y) + 1 = r(z)$, then we know that y and z are the (unique) strings in A that most tightly bracket x in the \leq and the $>$ directions. We can in our current case build the greatest string less than or equal to x that is in A bit by bit, querying potential prefixes, in polynomial time. Since $\text{rank}_A(x) = \text{rank}_A(y)$, we can compute $\text{rank}_A(x)$ in polynomial time for arbitrary x . ◀

From Proposition 4.3 and Theorem 4.8, we obtain the following corollary.

► **Corollary 4.9.** *If $P = \text{UP} \cap \text{coUP}$ then the complement of each nongappy semistrong-P-rankable set is strong-P-rankable (and so certainly is semistrong-P-rankable).*

► **Theorem 4.10.** *There exist P-rankable sets A and B such that $A \cap B$ is infinite but not F_{PR} -compressible.*

Proof. We will define a set A not containing the empty string and satisfying the condition that for all $x \in \Sigma^*$, exactly one of $x0$ and $x1$ is in A . Then clearly A is P-rankable by a compression function sending $x1$ and $x0$ to $\text{rank}_{\Sigma^*}(x)$. Let A_0 and B_0 be empty, and let $m_0 = \epsilon$. We will define A_i , B_i , and m_i inductively for $i > 0$. Let φ_i be the i th Turing machine in some enumeration of all Turing machines.

1. Suppose that φ_i is defined on $m_{i-1}0$, and that for all $x \in (A_{i-1} \cap B_{i-1}) \cup \{y \mid y > m_{i-1}0\}$ we have $\varphi_i(x) \neq \varphi_i(m_{i-1}0)$. In this case, we set $A_i = A_{i-1} \cup \{m_{i-1}0, \text{shift}(m_{i-1}, 1)0\}$ and $B_i = B_{i-1} \cup \{m_{i-1}1, \text{shift}(m_{i-1}, 1)0\}$ and set $m_i = \text{shift}(m_{i-1}, 2)$, so that neither $m_{i-1}0$ nor $m_{i-1}1$ is in $A_i \cap B_i$. Note that $\text{shift}(m_{i-1}, 1)0 \in A_i \cap B_i$ but $\text{shift}(m_{i-1}, 1)0 \notin A_{i-1} \cap B_{i-1}$.
2. Suppose φ_i is either undefined on $m_{i-1}0$, or that for some $x \in A_{i-1} \cap B_{i-1}$ we have $\varphi_i(x) = \varphi_i(m_{i-1}0)$. In this case, set $A_i = A_{i-1} \cup \{m_{i-1}0\}$, $B_i = B_{i-1} \cup \{m_{i-1}0\}$, and $m_i = \text{shift}(m_{i-1}, 1)$. Note in particular that x and $m_{i-1}0$ are both in $A_i \cap B_i$ and lexicographically less than m_i0 , and take the same value under φ_i .
3. Suppose that the above cases do not hold and there is some $x > m_{i-1}1$ such that $\varphi_i(x) = \varphi_i(m_{i-1}0)$. Let y be the lexicographically largest string such that $y0 \leq x$, and let $m_i = \text{shift}(y, 1)$. Set $A_i = A_{i-1} \cup \{z0 \mid m_{i-1} \leq z < y\} \cup \{x\}$ and $B_i = B_{i-1} \cup \{z0 \mid m_{i-1} \leq z < y\} \cup \{x\}$. Note in particular that x and $m_{i-1}0$ are both in $A_i \cap B_i$ and lexicographically less than m_i0 , and take the same value under φ_i .

Finally, let $A = \bigcup_{i \geq 0} A_i$ and $B = \bigcup_{i \geq 0} B_i$. Notice that stage i only adds elements to A_i or B_i that are lexicographically greater than or equal to $m_{i-1}0$, so if $x < m_i0$ and $x \notin A_i \cap B_i$, then $x \notin A \cap B$. In case 1, we see that φ_i fails to be surjective (i.e., onto Σ^*) when restricted to $A \cap B$, since there is no $x < m_i0$ in $A \cap B$ mapping to $\varphi_i(m_{i-1}0)$, and also no $x > m_{i-1}1$ mapping to $\varphi_i(m_{i-1}0)$, and neither $m_{i-1}0$ nor $m_{i-1}1$ is in $A \cap B$. In case 2, we see either that φ_i is undefined on an element of $A \cap B$ or that two elements of $A \cap B$ map to the same element. In case 3, we see that two elements in $A \cap B$ map to the same element under φ_i . Thus φ_i fails to compress $A \cap B$, and no partial recursive function can compress $A \cap B$. The set $A \cap B$ is infinite since at least one new element is added to $A_i \cap B_i$ during stage i . We also maintain the condition that, for all $x < m_i$, exactly one of $x0$ and $x1$ is in A_i (resp., B_i). Each A_i (resp. B_i) consists of exactly all strings in A (resp. B) lexicographically less than m_i0 , and so clearly since this statement holds for each A_i (resp. B_i) it holds for all of A (resp. B) as well. Thus A and B are P-rankable, but their intersection is not F_{PR} -compressible. ◀

► **Theorem 4.11.** *There exist infinite P-rankable sets A and B such that $A \cup B$ is not F_{PR} -compressible.*

► **Theorem 4.12.** *There exists an infinite P-rankable set whose complement is infinite but not F_{PR} -compressible.*

► **Theorem 4.13.** *There exist sets A and B that are not F_{PR} -compressible, yet $A \cup B$ is strong-P-rankable. In addition, there exist sets A and B that are not F_{PR} -compressible, yet $A \cap B$ is strong-P-rankable.*

The proofs of these three theorems are in the appendix.

5 Additional Closure and Nonclosure Properties

How robust are the polynomial-time and recursion-theoretically compressible and the rankable sets? Do sets lose these properties under join, or subtraction, addition, or (better yet) symmetric difference with finite sets? Or even with sufficiently nice infinite sets? The following section addresses these questions.

5.1 Complexity-Theoretic Results

We focus on the join (aka disjoint union), giving a full classification of the closure properties (or lack thereof) of the P-rankable, semistrong-P-rankable, and strong-P-rankable sets, as well as their complements, under this operation. The literature is not consistent as to whether the low-order or high-order bit is the “marking” bit for the join. Here, we follow the classic computability texts of Rogers [15] and Soare [17] and the classic structural-complexity text of Balcázar, Díaz, Gabarró [4], and define the join using low-order-bit marking: The join of A and B , denoted $A \oplus B$, is $A0 \cup B1$, i.e., $\{x0 \mid x \in A\} \cup \{x1 \mid x \in B\}$. For classes invariant under reversal, which end is used for the marking bit is not important (in the sense that the class itself is closed under upper-bit-marked join if and only if it is closed under lower-bit-marked join). However, the placement of the marking bit potentially matters for ranking-based classes, since those classes are based on lexicographical order.

The join is such a basic operation that it seems very surprising that any class would not be closed under it, and it would be even more surprising if the join of two sets that lack some nice organizational property (such as being P-rankable) can have that property (can be P-rankable, and we indeed show in this section that that happens)—i.e., the join of two sets can be “simpler” than either of them (despite the fact that the join of two sets is the least upper bound for them with respect to \leq_m^P [16], and in the sense of reductions captures the power-as-a-target of both sets). However, there is a precedent for this in the literature, and it regards a rather important complexity-theoretic structure. It is known that $(\text{EL}_2)^G$ is not closed under the join [10], where EL_2 is the second level of the extended low hierarchy [3].

► **Theorem 5.1.** *If $P \neq P^{\#P}$ then there exist sets $A \in P$ and $B \in P$ that are not P-rankable yet $A \cap B$, $A \cup B$, and $A \oplus B$ are strong-P-rankable.*

Proof. In this proof we construct a set A_1 whose members represent satisfying assignments of boolean formulas. When we force certain elements, or beacons, into A_1 we obtain a set A such that if we were able to rank A , we could count the number of satisfying assignments to a boolean formula by comparing the rank of these beacons. The set B is constructed similarly, but in a way that $A \cup B$, $A \cap B$, and $A \oplus B$ are easily strong-P-rankable.

As in the rest of the paper, $\Sigma = \{0, 1\}$. Let $A_1 = \{\alpha 01\beta \mid \alpha, \beta \in \Sigma^* \wedge |\alpha| = |\beta| \wedge \alpha \text{ is a valid encoding of boolean formula } F \text{ that has (without loss of generality) } k \leq |\alpha| \text{ variables, the first } k \text{ bits of } \beta \text{ encode a satisfying assignment of } F, \text{ and the rest of the } |\beta| - k \text{ bits of } \beta \text{ are } 0\}\}$. Note that given a string $x = \alpha 01\beta \in A_1$, we can unambiguously extract α and β because they must have length $(|x| - 2)/2$. Let $B_1 = \{\alpha 01\beta \mid \alpha, \beta \in \Sigma^* \wedge |\alpha| = |\beta| \wedge \alpha 01\beta \notin A_1\}$. Let $\text{Beacons} = \{\alpha 000^{|\alpha|} \mid \alpha \in \Sigma^*\} \cup \{\alpha 110^{|\alpha|} \mid \alpha \in \Sigma^*\}$. Similarly to A_1 , strings in B_1 and Beacons can be parsed unambiguously. Let $A = A_1 \cup \text{Beacons}$. Let $B = B_1 \cup \text{Beacons}$. Note that A and B are both in P because checking if an assignment satisfies a boolean formula is in P and Beacons is clearly in P .

We will now demonstrate that if either A or B were P-rankable, then $\#SAT$ would be in P . Suppose that A is P-rankable and let f be a polynomial-time ranking function

for A . Let α be a string encoding a boolean formula F . Then we can compute $j = f(\alpha 110^{|\alpha|}) - f(\alpha 000^{|\alpha|})$ in polynomial time. Both $\alpha 110^{|\alpha|}$ and $\alpha 000^{|\alpha|}$ are in *Beacons* and thus in A , so f gives a true ranking for these values. Every string in A between (and not including) these *Beacons* strings is from A_1 and thus represents a satisfying assignment for F , and every satisfying assignment for F is represented by a string between these *Beacons* strings. Because the last $|\beta| - k$ bits of β are 0, where k is the number of variables in F , each satisfying assignment for F is represented exactly once between the two *Beacons* strings. Thus $j - 1$ is the number of satisfying assignments of F . We can compute j in polynomial time, so $\#SAT$ is polynomial-time computable and thus $P = P^{\#P}$, contrary to our $P \neq P^{\#P}$ hypothesis.

Now suppose that B is P -rankable and similarly to before we will let f be the P -time ranking function for it. Again we will let α be the encoding for some boolean formula F and $j = f(\alpha 110^{|\alpha|}) - f(\alpha 000^{|\alpha|})$. In this case the strings in B between $\alpha 110^{|\alpha|}$ and $\alpha 000^{|\alpha|}$ are the strings of the form $\alpha 01\Sigma^{|\alpha|}$ except for those that are in A_1 (and recall that those that are in A_1 are precisely the padded-with-0s satisfying assignments for F). Because we know the number of strings of the form $\alpha 01\Sigma^{|\alpha|}$, we can again find the number of satisfying assignments for F . Namely, we have that $j = 1 + 2^{|\alpha|} - s$, where s is the number of satisfying assignments of F . Thus if B is P -rankable, then we can find s in polynomial time and thus $P = P^{\#P}$, contrary to our $P \neq P^{\#P}$ hypothesis.

Finally, we show that $A \cup B$, $A \cap B$, and $A \oplus B$ are strong- P -rankable. The set $A \cap B$ is simply *Beacons*, which is strong- P -rankable as follows. Any string lexicographically below 00 has rank 0. For any $\alpha \in \Sigma^*$, the rank of $\alpha 000^{|\alpha|}$ is $2\text{rank}_{\Sigma^*}(\alpha) - 1$ and the rank of $\alpha 110^{|\alpha|}$ is $2\text{rank}_{\Sigma^*}(\alpha)$. For every other string, it is easy to find the lexicographically greatest string in *Beacons* that is lexicographically less than the given string in polynomial time, and so it is possible to rank the string in polynomial time.

The set $A \cup B = \{\alpha 01\beta \mid \alpha \in \Sigma^* \wedge \beta \in \Sigma^* \wedge |\alpha| = |\beta|\} \cup \{\alpha 000^{|\alpha|} \mid \alpha \in \Sigma^*\} \cup \{\alpha 110^{|\alpha|} \mid \alpha \in \Sigma^*\}$, and is also strong- P -rankable, as follows. Any string lexicographically below 00 has rank 0. For any $\alpha \in \Sigma^*$, the rank of $\alpha 000^{|\alpha|}$ is $1 + \sum_{x <_{lex} \alpha} (2^{|x|} + 2)$, where $x <_{lex} \alpha$ denotes that x is lexicographically less than α . Note that although the sum is over an exponentially sized set, it still can be computed in polynomial time because the summands depend only on the length of the element in the set. Let $b(x)$ be the number of strings lexicographically less than α but with the same length as α . Then we have that $1 + \sum_{x <_{lex} \alpha} (2^{|x|} + 2) = 1 + b(\alpha)(2^{|\alpha|} + 2) + \sum_{i=0}^{|\alpha|-1} (2^i(2^i + 2))$.

The rank of $\alpha 110^{|\alpha|}$ is $\sum_{x \leq_{lex} \alpha} (2^{|x|} + 2)$, where $x \leq_{lex} \alpha$ denotes that x is lexicographically less than or equal to α . For any $\alpha, \beta \in \Sigma^*$ where $|\alpha| = |\beta|$, the rank of $\alpha 01\beta$ is $b(\beta) + 2 + \sum_{x <_{lex} \alpha} (2^{|x|} + 2)$, where n is the integer such that β is the n th string of its length. As above, each term is only dependent on the length of x , and is computable in polynomial time. For any other not string in $A \cup B$, it is easy to find the greatest string in $A \cup B$ lexicographically less than the given string in polynomial time, and thus it is easy to rank that string.

We can show that $A \oplus B = \{a0 \mid a \in A\} \cup \{b1 \mid b \in B\}$ is strong- P -rankable using the fact that $A \cup B$ and $A \cap B$ are strong- P -rankable, and both A and B are in P . The rank of ϵ is 0. The rank of 0 is 1 if $\epsilon \in A$ and otherwise is 0. For $x \in \Sigma^*$, we have $\text{rank}_{A \oplus B}(x1) = \text{rank}_{A \cup B}(x) + \text{rank}_{A \cap B}(x)$. For $x \neq \epsilon$, we have $\text{rank}_{A \oplus B}(x0) = \text{rank}_{A \oplus B}(x1) - \delta_B(x)$, where $\delta_B(x) = 1$ if and only if $x \in B$. ◀

► **Theorem 5.2.** *The following are equivalent:*

1. *strong- P -rankable^G is closed under join,*
2. *semistrong- P -rankable^G is closed under join, and*

3. $P = P^{\#P}$.

Proof. Theorem 5.1 shows that either of 1 or 2 would imply 3. Now we show that 3 implies 1 and 2, or equivalently the negation of either 1 or 2 would imply the negation of 3. Suppose that strong- P -rankable^G (resp., semistrong- P -rankable^G) is not closed under join. Then there are two sets A and B that are in strong- P -rankable^G (resp., semistrong- P -rankable^G) but $A \oplus B$ is strong- P -rankable (resp., semistrong- P -rankable^G). Then both A and B are in P . This is because $A \oplus B \in P$ and to test x for membership in A , for example, we can just test $x0$ for membership in $A \oplus B$. It was shown by Hemachandra and Rudich [9] that $P = P^{\#P}$, $P = \text{semistrong-}P\text{-rankable}$, and $P = \text{strong-}P\text{-rankable}$ are equivalent. Since A and B are in P but not strong- P -rankable (resp., semistrong- P -rankable), $P \neq \text{strong-}P\text{-rankable}$ (respectively $P \neq \text{semistrong-}P\text{-rankable}$) and thus $P \neq P^{\#P}$. ◀

► **Theorem 5.3.** *The class $P\text{-rankable}^G$ is not closed under join.*

► **Theorem 5.4.** *The class $P\text{-rankable}$ is not closed under join.*

► **Theorem 5.5.** *The class strong- P -rankable is closed under join.*

► **Theorem 5.6.** *The class semistrong- P -rankable is closed under complement if and only if it is closed under join.*

► **Theorem 5.7.** *The class $P\text{-compressible}'$ is closed under join.*

The proofs of Theorems 5.3–5.7 are in the appendix.

5.2 Recursion-Theoretic Results

- **Theorem 5.8.** 1. *If A is an F_{REC} -rankable set, $B_1 \subseteq A$ is a recursive set, and $B_2 \subseteq \overline{A}$ is a recursive set, then $A \triangle (B_1 \cup B_2)$ (equivalently, $(A - B_1) \cup B_2$) is F_{REC} -rankable.*
- 2. *If A is F_{REC} -compressible, $B_1 \subseteq A$ is recursive, and $A - B_1$ contains an infinite RE subset, then $A - B_1$ is F_{REC} -compressible.*
- 3. *If A is an F_{REC} -compressible set and $B_2 \subseteq \overline{A}$ is a recursive set, then $A \cup B_2$ is an F_{REC} -compressible set.*

Theorem 5.8's proof is in the appendix.

- **Corollary 5.9.** 1. *The class of F_{REC} -rankable sets is closed under symmetric difference with finite sets (and thus also under removing and adding finite sets).*
- 2. *The class of F_{REC} -compressible sets is closed addition and subtraction of finite sets.*

6 Conclusions

Taking to heart the work in earlier papers that views as classes the collections of sets that have (or lack) rankability/compressibility properties, we have studied whether those classes are closed under the most important boolean and other operations. For the studied classes, we in almost every case were able to prove that they are closed under the operation, or to prove that they are not closed under the operation, or to prove that whether they are closed depends on well-known questions about the equality of standard complexity classes. Additionally, we have introduced the notion of compression onto a set and have showed the robustness of compression under this notion, as well as the limits of that robustness. Appendix B provides some additional directions and some preliminary results on them.

References

- 1 E. Allender. Invertible functions, 1985. PhD thesis, Georgia Institute of Technology.
- 2 C. Álvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107:3–30, 1993.
- 3 J. Balcázar, R. Book, and U. Schöning. Sparse sets, lowness and highness. *SIAM Journal on Computing*, 15(3):739–746, 1986.
- 4 J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2nd edition, 1995.
- 5 A. Bertoni, M. Goldwurm, and N. Sabadini. The complexity of computing the number of strings of given length in context-free languages. *Theoretical Computer Science*, 86(2):325–342, 1991.
- 6 A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20(3):524–536, 1991.
- 7 J. Goldsmith, L. Hemachandra, and K. Kunen. Polynomial-time compression. *Computational Complexity*, 2(1):18–39, 1992.
- 8 J. Goldsmith and S. Homer. Scalability and the isomorphism problem. *Information Processing Letters*, 57(3):137–143, 1996.
- 9 L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41(2):251–271, 1990.
- 10 L. Hemaspaandra, Z. Jiang, J. Rothe, and O. Watanabe. Boolean operations, joins, and the extended low hierarchy. *Theoretical Computer Science*, 205(1–2):317–327, 1998.
- 11 L. Hemaspaandra and D. Rubery. Recursion-theoretic ranking and compression. *Journal of Computer and System Sciences*. To appear; preliminary version available as [12].
- 12 L. Hemaspaandra and D. Rubery. Recursion-theoretic ranking and compression. Technical Report arXiv:1606.01185 [cs.LO], Computing Research Repository, arXiv.org/corr/, October 2016. Revised, December 2017.
- 13 L. Hemaspaandra, M. Zaki, and M. Zimand. Polynomial-time semi-rankable sets. In *Journal of Computing and Information*, 2(1), Special Issue: *Proceedings of the 8th International Conference on Computing and Information*, pages 50–67, 1996. CD-ROM ISSN 1201-8511/V2/#1.
- 14 D. Huynh. The complexity of ranking simple languages. *Mathematical Systems Theory*, 23(1):1–20, 1990.
- 15 H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.
- 16 U. Schöning. *Complexity and Structure*. Springer-Verlag *Lecture Notes in Computer Science* #211, 1986.
- 17 R. Soare. *Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets*. Perspectives in Mathematical Logic. Springer-Verlag, 1987.

A

Appendix

In this section, we include proofs omitted from earlier sections.

Proof of Theorem 4.11. Let a set A not containing the empty string satisfy the condition that for all $x \in \Sigma^*$, exactly one of $x0$, $x1$ is in A . Then clearly A is P-rankable by a function sending $x1$ and $x0$ to $\text{rank}_{\Sigma^*}(x)$.

Let A_0 and B_0 be empty, and let $m_0 = \epsilon$. We will construct A_i , B_i , and m_i inductively for $i > 0$. Let φ_i be the i th Turing machine in some enumeration of all Turing machines.

1. Suppose that φ_i is defined on $m_{i-1}0$, and that for all x in $A_{i-1} \cup B_{i-1} \cup \{y \mid y > m_{i-1}0\}$ we have $\varphi_i(x) \neq \varphi_i(m_{i-1}0)$. In this case, we set $A_i = A_{i-1} \cup \{m_{i-1}1\}$ and $B_i = B_{i-1} \cup \{m_{i-1}1\}$, and we set $m_i = \text{shift}(m_{i-1}, 1)$.
2. Suppose φ_i is either undefined on $m_{i-1}0$, or that for some $x \in A_{i-1} \cup B_{i-1}$ we have $\varphi_i(x) = \varphi_i(m_{i-1}0)$. In this case, set $A_i = A_{i-1} \cup \{m_{i-1}0\}$, $B_i = B_{i-1} \cup \{m_{i-1}0\}$, and $m_i = \text{shift}(m_i, 1)$.
3. Suppose that the above cases do not hold and there is some $x \geq m_{i-1}0$ such that $\varphi_i(x) = \varphi_i(m_{i-1}0)$. Let m_i be the lexicographically smallest string such that $m_i0 > x$. Set $A_i = A_{i-1} \cup \{y0 \mid m_{i-1} \leq y < m_i\}$ and $B_i = B_{i-1} \cup \{y1 \mid m_{i-1} \leq y < m_i\}$. Note that both $m_{i-1}0$ and x are in $A_i \cup B_i$.

Finally, let $A = \bigcup_{i \geq 0} A_i$ and $B = \bigcup_{i \geq 0} B_i$. Stage i only adds elements to A_i or B_i that are lexicographically greater than or equal to $m_{i-1}0$, so if $x < m_i0$ and $x \notin A_i \cup B_i$, then $x \notin A \cup B$. In case 1, we see that φ_i fails to be surjective (i.e., onto Σ^*) when restricted to $A \cup B$, since there is no $x < m_{i-1}0$ in $A \cup B$ mapping to $\varphi_i(m_{i-1}0)$, and also no $x > m_i0$ mapping to $\varphi_i(m_{i-1}0)$, so no element in $A \cup B$ compresses to $\varphi_i(m_{i-1}0)$. In case 2, we see either that φ_i is undefined on $m_{i-1}0 \in A \cup B$ or that $m_{i-1}0$ and some other element in $A \cup B$ map to the same element, so injectivity when restricted to $A \cap B$ fails. Similarly, in case 3, we see that $m_{i-1}0$ and some other element in $A \cup B$ will map to the same value under φ_i . Thus for all i we see that φ_i fails to compress $A \cup B$, and so no partial recursive function compresses $A \cup B$. Note that we maintain the condition that for all $x < m_i$, exactly one of $x0$ and $x1$ is in A_i (resp., B_i). This condition holds in A (resp., B), and this property carries over to A and B as well. Each A_i (resp., B_i) consists of exactly all strings in A (resp., B) lexicographically less than m_i0 , and so clearly since this statement holds for each A_i (resp., B_i) it holds for all of A (resp., B) as well. Thus A and B are P-rankable, but $A \cup B$ is not FPR -compressible. \blacktriangleleft

Proof of Theorem 4.12. We will construct a set A consisting of strings with length at least 2, with the property that for every $x \in \Sigma^*$, exactly one of $x00$, $x01$, $x10$, and $x11$ is in A . Clearly A will be infinite, and its complement is infinite as well. Also, A will be P-rankable by sending $x00$, $x01$, $x10$ and $x11$ to $\text{rank}_{\Sigma^*}(x)$. Let $A_0 = 0$ and $m_0 = \epsilon$. We will construct A_i and m_i inductively for $i > 0$. Let φ_i be the i th Turing machine in some enumeration of all Turing machines.

1. Suppose φ_i halts on $m_{i-1}00$, and there is no $x \in \overline{A_{i-1}}$ where $x < m_{i-1}00$ such that $\varphi_i(x) = \varphi_i(m_{i-1}00)$, and that there is no $x > m_{i-1}00$ such that $\varphi_i(x) = \varphi_i(m_{i-1}00)$. Then set $A_i = A_{i-1} \cup \{m_{i-1}00\}$ and set $m_i = \text{shift}(m_{i-1}, 1)$.
2. Suppose φ_i is undefined on $m_{i-1}00$, or that there is some $x < m_{i-1}00$ where $x \in \overline{A_{i-1}}$ and $\varphi(x) = \varphi_i(m_{i-1}00)$. Then set $A_i = A_{i-1} \cup \{m_{i-1}01\}$ and set $m_i = \text{shift}(m_{i-1}, 1)$.
3. Suppose that the above cases do not hold, φ_i is defined on $m_{i-1}00$, and $\varphi_i(m_{i-1}00) = \varphi_i(x)$ for some $x \in \{m_{i-1}01, m_{i-1}10, m_{i-1}11\}$. Then set $A_i = A_{i-1} \cup \{z\}$, where z is a fixed arbitrary element in $\{m_{i-1}01, m_{i-1}10, m_{i-1}11\} - \{x\}$, and set $m_i = \text{shift}(m_{i-1}, 1)$. Note that $m_{i-1}00$ and x are both in \overline{A} and below m_i00 , and take the same value under φ_i .
4. Suppose the above cases do not hold and φ_i is defined on $m_{i-1}00$ and $\varphi_i(m_{i-1}00) = \varphi_i(x)$ for some $x > m_i11$. Let y be equal to x without its last two characters, and set $m_i = \text{shift}(y, 1)$. Let $A_i = A_{i-1} \cup \{m_{i-1}01\} \cup \{z11 \mid m_i < z < y\} \cup \{w\}$, where w is some element in $\{y00, y01, y10, y11\} - \{x\}$. Note that $m_{i-1}00$ and x are both in $\overline{A_i}$ and lexicographically less than m_i00 , and take the same value under φ_i .

Finally, let $A = \bigcup_{i \geq 0} A_i$. Notice that stage i adds to A only elements that are lexicographically greater than or equal to $m_{i-1}00$, so if $x < m_i00$ and $x \in \overline{A_i}$, then $x \in \overline{A}$. In case 1, we see that φ_i fails to be surjective (i.e., onto Σ^*) when restricted to \overline{A} , since there is no $x < m_i00$ in \overline{A} mapping to $\varphi_i(m_{i-1}00)$, and also no $x \geq m_i0$ mapping to $\varphi_i(m_{i-1}00)$. In case 2, either φ_i does not halt on $m_i00 \in \overline{A}$ or there are two elements in \overline{A} that take the same value under φ_i . In cases 3 and 4, we see that there are two elements in \overline{A} that take the same value under φ_i . Thus in all cases φ_i fails to compress \overline{A} and so \overline{A} is not F_{PR} -compressible. \blacktriangleleft

Proof of Theorem 4.13. Let C be a set such that $A = C0 \cup \Sigma^*1$ is not F_{PR} -compressible. Such a set can be constructed using a similar method to those of Theorems 4.11, 4.12, and 4.13. Then $B = C1 \cup \Sigma^*0$, $A' = C00 \cup \Sigma^*1$, and $B' = C10 \cup \Sigma^*1$ are all also not F_{PR} -compressible, since clearly they are all recursively isomorphic. Note that $A \cup B = \Sigma^*$ and $A' \cap B' = \Sigma^*1$, both of which are strong-P-rankable. \blacktriangleleft

Proof of Theorem 5.3. Let A be any language that is not P-rankable and whose complement is not P-rankable. An example of such a set is $A = \{x000 \mid x \in \Sigma^*\} \cup \{x001 \mid x \in B\} \cup \{x010 \mid x \in \Sigma^*\} \cup \{x100 \mid x \in B\}$, where B is any undecidable set. This set is not even F_{REC} -rankable. Note $x \in B$ if and only if $\text{rank}_A(x010) - \text{rank}_A(x000) > 1$, so if A were F_{REC} -rankable, we could decide B . Similarly, $x \notin B$ if and only if $\text{rank}_{\overline{A}}(x101) - \text{rank}_{\overline{A}}(x011) > 1$, so if \overline{A} were F_{REC} -rankable, B would be decidable, but this is a contradiction.

Then $A \oplus \overline{A} = A0 \cup \overline{A}1$ is P-rankable. It can be ranked by any function mapping $x0$ and $x1$ to $\text{rank}_{\Sigma^*}(x)$. \blacktriangleleft

Proof of Theorem 5.4. Let A be some undecidable set. Let $A' = A \oplus \overline{A}$. Then A' is P-rankable by any function mapping $x0$ and $x1$ to $\text{rank}_{\Sigma^*}(x)$. Now let $B = \Sigma^* \oplus A'$. Then B is the join of two P-rankable sets. Suppose B were P-rankable, then we can query $\text{rank}_B(x0)$ for all strings x . If $\text{rank}_B(x0) + 2 = \text{rank}_B(\text{shift}(x, 1)0)$, we know that $x1 \in B$, and thus $x \in A'$. Otherwise, $x1 \notin B$ so $x1 \notin A'$. Since we can test membership in A' , we can test membership of x in A by asking whether $x0 \in A'$. This is a contradiction as A was assumed undecidable; thus B cannot be P-rankable. \blacktriangleleft

Proof of Theorem 5.5. Let A and B be strong-P-rankable. The rank of $x0$ in $A \oplus B$ is $\text{rank}_A(x) + \text{rank}_B(\text{shift}(x, -1))$, and the rank of $x1$ is $\text{rank}_A(x) + \text{rank}_B(x)$. The rank of ϵ is 0. All of these values can clearly be computed in polynomial time so $A \oplus B$ is strong-P-rankable. \blacktriangleleft

Proof of Theorem 5.6. Suppose semistrong-P-rankable is closed under complement. Then semistrong-P-rankable is equal to strong-P-rankable, so semistrong-P-rankable is closed under join.

Now suppose semistrong-P-rankable is closed under join. Let set A be semistrong-P-rankable by ranking function h . Let $X = \Sigma^* \oplus A$. Then X is the join of two semistrong-P-rankable sets and thus is semistrong-P-rankable by some ranking function f . The ranking function for \overline{A} does the following. Given x , if $h(x)$ returns a rank (rather than an indication that $x \notin A$) then return an indication that $x \notin \overline{A}$. Otherwise let $y = \text{shift}(x, 1)$ and return $2\text{rank}_{\Sigma^*}(x) + 1 - f(y0)$. There are a total of $2\text{rank}_{\Sigma^*}(x) + 2$ strings lexicographically less than or equal to $y0$ in Σ^* . All those missing in X correspond to either ϵ or strings not in A that are strictly less than y . Since $y0 \in X$, we know that $f(y0)$ of these are in X . The rest are in $\Sigma^* - X = \{x1 \mid x \in \overline{A}\} \cup \{\epsilon\}$. Thus the number of strings in \overline{A} below x

is $2\text{rank}_{\Sigma^*}(x) + 1 - f(y_0)$. Thus \overline{A} is semistrong-P-rankable, so semistrong-P-rankable is closed under complement. \blacktriangleleft

Proof of Theorem 5.7. Let A and B be two P-compressible' sets. Let f and g be the compression functions for A and B respectively. Let $h(x0) = f(x)0$ and $h(x1) = \text{shift}(f(x)0, -1)$. Now h is a compression function for $A \oplus B$ since the image of h restricted to $A0$ is Σ^*0 , and the image of h restricted to $B1$ is $\Sigma^*1 \cup \{\epsilon\}$. Each of $A0$ and $B0$ maps injectively because f and g are compression functions, and together they map injectively on all of $A \oplus B$ to all of Σ^* . The function h is clearly polynomial time, and so $A \oplus B$ is P-compressible'. \blacktriangleleft

Proof of Theorem 5.8. For the first part of this theorem, let f be an F_{REC} -ranking function for A . Since B_1 and B_2 are recursive, their ranking functions rank_{B_1} and rank_{B_2} are in F_{REC} . Our F_{REC} ranking function for $A \triangle (B_1 \cup B_2)$ is $f'(x) = f(x) + \text{rank}_{B_2}(x) - \text{rank}_{B_1}(x)$. This directly accounts for the additions and deletions done by B_1 and B_2 .

We now prove the second part of the theorem. The statement is clearly true if B_1 is finite, even in the case that $A - B_1$ does not contain an infinite RE subset (as long as $A - B_1$ is still infinite). This is because the image of $A - B_1$ under a compression function for A is cofinite, and cofinite sets are compressible. Thus composing a compression function for the cofinite image of $A - B_1$ with a compression function for A , we obtain a compression function for $A - B_1$.

So from this point on we assume that B_1 is infinite. Let h be an F_{REC} -compression function for A . By the hypothesis of the theorem, there is an infinite RE subset of $A - B_1$, call it C . Since every infinite RE set contains an infinite recursive subset, let $B_2 \subseteq C$ be infinite and recursive. Let $b_1 < b_2 < b_3 < \dots$ be the elements in B_1 , and let $c_1 < c_2 < c_3 < \dots$ be the elements in B_2 . Consider the following function.

$$g(x) = \begin{cases} x & \text{if } x \notin B_1 \cup B_2, \\ \epsilon & \text{if } x \in B_1, \\ b_{\lceil i/2 \rceil} & \text{if } x = c_i \text{ and } i \text{ is odd, and} \\ c_{i/2} & \text{if } x = c_i \text{ and } i \text{ is even.} \end{cases}$$

Let $f(x) = h(g(x))$. We claim that f is a compression function for $A - B_1$. We do this by showing g is a compression function for $A - B_1$ onto A , since we already know that h compresses A to Σ^* . See that g is the identity on $A - (B_1 \cup B_2)$. See also that $g(B_2) = B_1 \cup B_2$ injectively and surjectively. Since $(A - B_1) - B_2$ and B_2 are disjoint and have disjoint images, and since g is injective and surjective on both these domains onto their respective images, it follows that g is injective and surjective on $A - B_1$ to the image $g((A - B_1) - B_2) \cup g(B_2) = A$. Thus g is a compression function for $A - B_1$ to A , and h is a compression function for A to Σ^* , so f is a compression function for $A - B_1$ to Σ^* . In other words, $A - B_1$ is F_{REC} -compressible.

We now prove the third part of the theorem. Let f be an F_{REC} -compression function for A . If B_2 is finite, our F_{REC} compression function for $A \cup B_2$ is $f'(x) = \text{shift}(f(x), \|B_2\|)$ for $x \notin B_2$ and $f'(x) = \text{shift}(\epsilon, \text{rank}_{B_2}(x))$ for $x \in B_2$.

On other hand, if B_2 is infinite, let g be an F_{REC} compression function for B_2 , e.g., g can be taken to be (recall that B_2 is recursive) defined by $g(x)$ being the $\max(\text{rank}_{B_2}(x), 1)$ -st string in Σ^* . We define $f'(x)$ as follows. (Recall that for us Σ is always fixed as being $\{0, 1\}$.) If $x \notin B_2$ then $f'(x) = 1f(x)$ (i.e., $f(x)$ prefixed with a one). If $x \in B_2$ and $g(x) = \epsilon$ then $f'(x) = \epsilon$. And, finally, if $x \in B_2$ and $g(x) \neq \epsilon$ then $f'(x) = 0\text{shift}(g(x), -1)$. (The shift-by-one treatment of the $x \notin B$ case is because we must ensure that ϵ is mapped to

by some string in $A \cup B_2$.) Now, f' maps $A \cup B$ bijectively onto Σ^* , so f' is an F_{REC} compression function for $A \cup B$, so $A \cup B$ is F_{REC} -compressible. ◀

B Appendix

B.1 Relativization

The results of [11] all relativize in a straightforward manner. In this section, we include a few examples. This justifies our limitation to F_{REC} and F_{PR} : By relativization, we get analogous results about more powerful function classes, such as F_{Δ_2} .⁴

► **Theorem B.1.** *For each $i \geq 1$, $\Delta_i = \Sigma_i \cap F_{\Delta_i}$ -compressible'.*

Proof. Relativization of [12, Theorem 5.3] (see also [7, 11]). ◀

Since, for $i \geq 1$ $F_{\Delta_i} \supseteq F_{\text{REC}}$, we get the following easy corollary.

► **Corollary B.2.** *For each $i \geq 1$, $\Sigma_i \cap F_{\text{REC}}$ -compressible' $\subseteq \Delta_i$.*

► **Theorem B.3.** *For each $i \geq 1$, $\Pi_i \cap F_{\Delta_i}$ -rankable = $\Pi_i \cap F_{\text{PR}}^{\Sigma_{i-1}}$ -rankable.*

Proof. Relativization of [12, Theorem 4.6] (see also [11]). ◀

B.2 Compressibility, Honesty, and Selectivity

If we restrict our attention to honest functions, we can prove some very clean results. There is a little subtlety here, since there are many nonequivalent definitions of honesty. We use the following:

A (possibly partial) function f is honest on B if there is a recursive function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that for any $x \in \text{domain}(f) \cap B$, $g(|f(x)|) \geq |x|$. If f is honest on Σ^* , we say f is honest.

This gives two potential ways to define honest compressibility. For a given set A , we can require the compression function to be honest on Σ^* , or only on A . We call the former notion honestly- F -compressible, and the latter honestly-on- A - F -compressible. The following theorem asserts that these two notions are equivalent for F_{PR} and F_{REC} functions.

- **Theorem B.4.** 1. *For each set A , A is honestly- F_{PR} -compressible if and only if A is honestly-on- A - F_{PR} -compressible.*
 2. *For each set A , A is honestly- F_{REC} -compressible if and only if A is honestly-on- A - F_{REC} -compressible.*

Proof. The “only if” direction is trivial, since every honest on Σ^* function is honest on A .

For the “if” direction, let f be an honest on A compression function for A , and let g be a recursive honesty-bound function for f . Define f' as follows, where $\text{domain}(f') = \text{domain}(f)$. Over the domain of f , if $g(|f(x)|) \geq |x|$, then $f'(x) = f(x)$. If $g(|f(x)|) < |x|$, let $f'(x) = x$. Since f was honest on A , for any $x \in A$, $f'(x) = f(x)$. Thus f' is still a compression function for A . The recursive function $g'(n) = \max(g(n), n)$ satisfies, for all $x \in \text{domain}(f')$, $g'(|f'(x)|) \geq |x|$, and thus proves that f' is honest (on Σ^*).

Note that when f is recursive, so is f' , giving us the second part of the theorem. ◀

⁴ $F_{\Delta_{i+1}}$ will denote the class of total functions computed by Turing machines given access to a Σ_i oracle. Equivalently, $F_{\Delta_{i+1}} = (F_{\text{REC}})^{\Sigma_i}$. Note that $F_{\text{REC}} = F_{\Delta_1}$. The class of partial functions computed by Turing machines given access to a Σ_i oracle will be denoted $(F_{\text{PR}})^{\Sigma_i}$ or simply as $F_{\text{PR}}^{\Sigma_i}$.

The following proof uses F -selectivity, which was very rarely useful. A set A is F -selective if there is a function $f \in F$ of two arguments such that the following hold:

1. For any $x, y \in \Sigma^*$, either $f(x, y) = x$ or $f(x, y) = y$.
2. If $x \in A$ or $y \in A$, $f(x, y) \in A$.

Intuitively, f selects the “more likely” of its two inputs. When $x, y \notin A$, or $x, y \in A$, f can choose either input. It’s only restricted when one input is in A , and the other is not. Both F_{REC} -selectivity and honestly- F_{REC} -compressibility are fairly strong claims. Only the infinite recursive sets satisfy both. Let INFINITE denote the infinite sets over the alphabet Σ .

► **Theorem B.5.** $F_{\text{REC}}\text{-selective} \cap \text{honestly-}F_{\text{REC}}\text{-compressible} = \text{REC} \cap \text{INFINITE}$.

Proof. Every infinite recursive set is easily F_{REC} -selective and honestly- F_{REC} -compressible giving the \supseteq inclusion.

For the \subseteq inclusion, let A be honestly- F_{REC} -compressible by f , with honesty bound g , and let h be a F_{REC} selector function for A . Then, for any z , by the definition of compressibility and honesty:

$$\|\{w \mid f(w) = z \wedge |w| \leq g(|z|)\} \cap A\| = 1.$$

So define the finite set $Q_z = \{w \mid f(w) = z \wedge |w| \leq g(|z|)\}$. We know this set contains exactly one element of A , and this will allow us to decide A .

On input x , compute $f(x)$ and $Q_{f(x)}$. Then use the selector function to find the unique element $y \in Q_{f(x)}$ such that for any $z \in Q_{f(x)}$, $h(y, z) = y$. Such a y exists and is unique because there is exactly one element of A in $Q_{f(x)}$.

If $x = y$, then $x \in A$. Otherwise, $x \notin A$, so A is recursive. Since A was compressible, it is infinite as well. ◀

In fact, honestly- F_{REC} -compressible is much stronger than F_{REC} -compressible. While all coRE cylinders are F_{REC} -compressible (see [11]), no set in $\text{RE} - \text{REC}$ is honestly- F_{REC} -compressible. This was first stated, without proof, in the conclusion section of [7].

► **Theorem B.6** (See [7]). $\text{honestly-}F_{\text{REC}}\text{-compressible} \cap \text{coRE} = \text{REC} \cap \text{INFINITE}$.

Proof. Every infinite recursive set is easily coRE and honestly- F_{REC} -compressible giving the \supseteq inclusion.

For the \subseteq inclusion, let A be coRE and honestly- F_{REC} -compressible by a compression function f . Let M accept \overline{A} . Define the sets Q_z from the proof of Theorem B.5.

Then for any input x , compute $f(x)$ and $Q_{f(x)}$. Then dovetail applying M to each element of $Q_{f(x)}$ until only one remains. If the remaining element is x , then $x \in A$. Otherwise, $x \notin A$, so A is recursive and infinite. ◀

This next group of theorems builds to a result that if A is nonrecursive, F_{REC} -selective and F_{REC} -compressible then \overline{A} has an infinite RE subset. Since F_{REC} -selectivity is such a strong assumption, this theorem is of limited use. However, the arguments used to show it may prove useful in the proof of other claims.

► **Theorem B.7.** If A is F_{REC} -compressible via f and $f(\overline{A})$ is finite, then A is recursive.

Proof. Using the definition of compressibility, $L = \{x \in A \mid f(x) \in f(\overline{A})\}$ is finite. By the assumptions of the theorem, so is $f(\overline{A})$. But, $x \in A$ if and only if $x \in L \vee f(x) \notin f(\overline{A})$. Since both of these sets are finite, this condition is recursive, and so is A . ◀

Now we consider the case where $f(\overline{A})$ is infinite.

► **Theorem B.8.** *If A is F_{REC} -compressible via f and $f(\overline{A})$ is infinite, then there is an infinite RE set $B_A = \{(p_1, q_1), (p_2, q_2), \dots\}$ such that no string appears in more than one pair and each pair contains at least one element of \overline{A} .⁵*

Proof. We describe a machine that enumerates the desired set.

Initialize $Q = \emptyset$. Begin running $f(\epsilon), f(0), f(1), \dots$ in sequence. Since $f(\overline{A})$ is infinite, there will be two strings $x, y \notin Q$ where $f(x) = f(y)$. Enumerate (x, y) , and add both x and y to Q . The enumerated set will have the desired properties. ◀

► **Corollary B.9.** *If A is nonrecursive, F_{REC} -selective, and F_{REC} -compressible, then \overline{A} has an infinite RE subset.*

Proof. Create the set from Theorem B.8, and apply the F_{REC} -selector to each pair. If the selector chooses p_i , then $q_i \in \overline{A}$, and vice versa. So we can enumerate an infinite RE subset of \overline{A} by enumerating the elements not chosen by the selector. ◀

⁵ In the theorem and the proof (and similarly regarding the proof of the corollary) we should, to be formally correct, define and work with the one-dimensional set $\{\langle a, b \rangle \mid (a, b) \in B_A\}$, where $\langle \cdot, \cdot \rangle$ is a nice, standard pairing function; but let us consider that implicit.