Reductions between Disjoint NP-Pairs

Christian Glaßer[*]
Lehrstuhl für Informatik IV
Universität Würzburg, 97074 Würzburg, Germany

Alan L. Selman[†]
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260

Samik Sengupta[‡]
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260

April 21, 2003

Abstract

We prove that all of the following assertions are equivalent: There is a many-one complete disjoint NP-pair; there is a strongly many-one complete disjoint NP-pair; there is a Turing complete disjoint NP-pair such that all reductions are smart reductions; there is a complete disjoint NP-pair for one-to-one, invertible reductions; the class of all disjoint NP-pairs is uniformly enumerable.

Let A, B, C, and D be nonempty sets belonging to NP. A *smart* reduction between the disjoint NP-pairs (A, B) and (C, D) is a Turing reduction with the additional property that if the input belongs to $A \cup B$, then all queries belong to $C \cup D$. We prove under the reasonable assumption UP $\cap$ co-UP has a P-bi-immune set that there exist disjoint NP-pairs (A, B) and (C, D) such that (A, B) is truth-table reducible to (C, D), but there is no smart reduction between them. This paper contains several additional separations of reductions between disjoint NP-pairs.

We exhibit an oracle relative to which DisjNP has a truth-table-complete disjoint NP-pair, but has no many-one-complete disjoint NP-pair.

[*]Research performed at the University at Buffalo with support by a postdoctoral grant from the German Academic Exchange Service (Deutscher Akademischer Austauschdienst – DAAD). Email: glasser@informatik.uni-wuerzburg.de

[†]Research partially supported by NSF grant CCR-0307077. Email: selman@cse.buffalo.edu

[‡]Email: samik@cse.buffalo.edu

1 Introduction

Disjoint NP-pairs relate naturally to the existence of public-key cryptography [GS88] and relate closely to the theory of proof systems for propositional calculus [Raz94, Pud01]. In both areas, reductions between disjoint NP-pairs arise naturally. In particular, Razborov [Raz94] proved that existence of an optimal proof system implies existence of a many-one complete disjoint NP-pair. Köbler, Messner, and Torán [KMT03] defined a stronger form of many-one reduction. They state "The reduction considered (by Razborov) is a weak form of many-one reducibility ... we can improve the mentioned result showing that under assumption that TAUT has an optimal proof system, the class of disjoint NP-pairs has a complete pair with respect to the following stronger notion of many-one reducibility." In this paper, we prove that there exists a complete pair with respect to the "stronger notion" of many-one reducibility if and only if there exists a complete pair with respect to the "weak form". Thus, the results of Razborov and of Köbler, Messner, and Torán are equivalent. Nevertheless, it is apparently true that the "stronger notion" really is stronger. This is easy to see if we permit disjoint NP-pairs of the form (A, B) where either A or B can be finite sets. However, for disjoint NP-pairs whose components are infinite and coinfinite, we prove that the "stronger notion" is identical to the "weak form" if and only if $\mathrm{P} = \mathrm{NP}$.

We prove under reasonable hypothesis existence of two disjoint NP-pairs (A, B) and (C, D) such that there is no smart reduction from (A, B) to (C, D), even though (A, B) is truth-table reducible to (C, D). A *smart* reduction is a Turing reduction with the additional property that if the input belongs to $A \cup B$, then all queries belong to $C \cup D$. Grollmann and Selman [GS88] defined smart reductions in order to analyze a conjecture of Even *et al.* [ESY84]. In addition to these separations, we prove under reasonable hypothesis that truth-table reductions differ from bounded-truth-table reductions and that Turing reductions differ from truth-table reductions.

Now let us return to the discussion in the first paragraph, for we prove much more than the two equivalent assertions we discussed there. Namely, we prove that all of the following assertions are equivalent:

- There is a many-one complete disjoint NP-pair.
- There is a many-one complete disjoint NP-pair using the "stronger notion."
- There is a Turing complete disjoint NP-pair such that all reductions are smart reductions.
- There is a complete disjoint NP-pair for one-to-one, invertible reductions.
- The class of all disjoint NP-pairs is uniformly enumerable.

There is a long history of equating having complete sets with uniform enumerations. Hartmanis and Hemachandra, for example, proved this for the class UP, and it holds as well for $\mathrm{NP} \cap \text{co-NP}$ and BPP. More recently, Sadowski [Sad02] proved that there exists an optimal propositional proof system if and only if the class of all easy subsets of TAUT is uniformly enumerable.

It follows from the previous paragraph that the following open questions are equivalent:

1. Does existence of a Turing-complete disjoint NP-pair imply existence of a many-complete disjoint NP-pair?

2. Does existence of a Turing-complete disjoint NP-pair imply existence of a smart Turing-complete disjoint NP-pair?

We address these open questions to the extent that we construct an oracle relative to which there exists a truth-table complete disjoint NP-pair while no disjoint NP-pair is many-one complete. Therefore, if the open question has a positive answer, no proof can relativize to all oracles.

2 Preliminaries

We fix the alphabet $\Sigma = \{0, 1\}$ and we denote the length of a word w by $|w|$. The set of all words is denoted by Σ^*. For a set of words X, let $X^{<n} \overset{df}{=} X \cap \Sigma^{<n}$, and define $X^{\leq n}$, $X^{=n}$, $X^{\geq n}$, and $X^{>n}$ analogously. For sets of words we take the complement with respect to Σ^*. The set of (nonzero) natural numbers is denoted by $\mathbb{N}$ (by $\mathbb{N}^+$, respectively). We use polynomial-time-computable and polynomial-time invertible pairing functions $\langle \cdot, \cdot \rangle : \mathbb{N}^+ \times \mathbb{N}^+ \rightarrow \mathbb{N}^+$.

We fix the following enumerations: $\{N_i\}_i$ is an effective enumeration of nondeterministic, polynomial-time-bounded Turing machines, $\{M_i\}_i$ is an effective enumeration of deterministic, polynomial-time-bounded oracle Turing machines, and $\{f_i\}_i$ is an effective enumeration of polynomial-time-computable functions. Moreover, $n^i + i$ is the running time for M_i (for any oracle), N_i, and f_i on inputs of length n. We can assume that given the code of a machine N, we can determine the index i such that $N = N_i$ in polynomial time in the length of the code. Furthermore, given i, we can determine the code of the machine in time polynomial in $|i|$.

Definition 2.1 *A* disjoint NP-pair *is a pair of nonempty sets A and B such that $A, B \in \mathrm{NP}$ and $A \cap B = \emptyset$. Let* DisjNP *denote the class of all disjoint* NP*-pairs.*

Definition 2.2 DisjNP *is* uniformly enumerable *if there is a total computable function $f : \Sigma^* \rightarrow \Sigma^* \times \Sigma^*$ such that*

1. $\forall (i, j) \in \mathrm{range}(f)[(L(M_i), L(M_j)) \in \mathrm{DisjNP}]$.

2. $\forall (C, D) \in \mathrm{DisjNP}\ \exists (i, j)[(i, j) \in \mathrm{range}(f) \wedge C = L(M_i) \wedge D = L(M_j)]$.

Given a disjoint NP-pair (A, B), a *separator* is a set S such that $A \subseteq S$ and $B \subseteq \overline{S}$; we say that S *separates* (A, B). Let $Sep(A, B)$ denote the class of all separators of (A, B). For disjoint NP-pairs (A, B), a fundamental question is whether $Sep(A, B)$ contains a set belonging to P. In that case the pair is P-*separable*; otherwise, the pair is P-*inseparable*.

Definition 2.3 *Let (A, B) be a disjoint* NP*-pair. $X \leq_T^{pp} (A, B)$ if for every separator S of (A, B), $X \leq_T^p S$.*

We define the standard reductions between disjoint pairs. Here we give the uniform versions. See Grollmann and Selman [GS88] and Glaßer et. al [GSSZ03] for the equivalences with the non-uniform versions.

Definition 2.4

1. (C, D) is many-one reducible to (A, B) in polynomial time, $(C, D) \leq_m^{pp} (A, B)$, *if there is a polynomial-time computable total function* f *such that* $f(C) \subseteq A$ *and* $f(D) \subseteq C$. *We also say that* $(C, D) \leq_m^{pp} (A, B)$ *via* f.

2. (C, D) is Turing reducible to (A, B) in polynomial time, $(C, D) \leq_T^{pp} (A, B)$, *if there exists a polynomial-time oracle Turing machine* M *such that for every separator* $T \in Sep(A, B)$, *there exists a separator* $S \in Sep(C, D)$ *such that* $S \leq_T^p T$ *via* M.

3. (C, D) is truth-table reducible to (A, B) in polynomial time, $(C, D) \leq_{tt}^{pp} (A, B)$, *if there exists a polynomial-time oracle Turing machine* M *such that for every separator* $T \in Sep(A, B)$, *there exists a separator* $S \in Sep(C, D)$ *such that* $S \leq_{tt}^p T$ *via* M.

4. (C, D) is bounded-truth-table reducible to (A, B) in polynomial time, $(C, D) \leq_{btt}^{pp} (A, B)$, *if there exists a polynomial-time oracle Turing machine* M *such that for every separator* $T \in Sep(A, B)$, *there exists a separator* $S \in Sep(C, D)$ *such that* $S \leq_{btt}^p T$ *via* M.

For pairs in DisjNP, we define the following version of invertible reductions.

Definition 2.5 (C, D) is many-one reducible to (A, B) in polynomial time via invertible reductions, $(C, D) \leq_{1\text{-}i}^{pp} (A, B)$, *if there are polynomial-time-computable total functions* f *and* g *such that* f *is one-to-one,* $(C, D) \leq_m^{pp} (A, B)$ *via* f, *and for every* x, $g(f(x)) = x$.

Strongly many-one reductions are defined by Köbler et al. [KMT03].

Definition 2.6 ([KMT03]) (C, D) strongly many-one reduces to (A, B) in polynomial time, $(C, D) \leq_{sm}^{pp} (A, B)$, *if there is a polynomial-time-computable total function* f *such that* $f(C) \subseteq A$, $f(D) \subseteq B$, *and* $f(\overline{C \cup D}) \subseteq \overline{A \cup B}$. *We also say that* $(C, D) \leq_{sm}^{pp} (A, B)$ *via* f.

It is easy to see that if $(C, D) \leq_{sm}^{pp} (A, B)$ via f, then $C \leq_m^p A$ via f, and $D \leq_m^p B$ via f.

Smart reductions are defined by Grollmann and Selman [GS88] and Goldreich and Goldwasser [GG98].

Definition 2.7 ([GS88, GG98]) *A* smart reduction *from* (C, D) *to* (A, B) *is a Turing reduction from* (C, D) *to* (A, B) *such that if the input belongs to* $C \cup D$, *then all queries belong to* $A \cup B$.

A disjoint pair $(A, B) \in$ DisjNP is *smart* $\leq_T^{pp}$*-complete* for DisjNP if for every (C, D) in DisjNP there is a smart reduction from (C, D) to (A, B). Note that if (A, B) is $\leq_m^{pp}$-complete for DisjNP, then (A, B) is smart $\leq_T^{pp}$-complete for DisjNP as well.

A language L is *immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$*-immune*, if L is infinite and no infinite subset of L belongs to $\mathcal{C}$. A language L is *bi-immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$*-bi-immune*, if both L and $\overline{L}$ are $\mathcal{C}$-immune.

A nondeterministic transducer T computes a value y on an input x if there is an accepting computation of T on x for which y is the final contents of the output tape of T. Such transducers compute partial multivalued functions.

Definition 2.8 ([BLS84, Sel94])

1. NPMV *is the set of all partial, multivalued functions computed by nondeterministic polynomial-time-bounded transducers.*

2. NPSV *is the set of all $f \in$* NPMV *that are single-valued.*

Following Köbler and Messner [KM00], we denote the class of all 0,1-valued functions in NPSV by $\mathrm{NPSV}_{\{0,1\}}$.

Given partial multivalued functions f and g, define g to be a *refinement* of f if $\mathrm{domain}(g) = \mathrm{domain}(f)$ and for all $x \in \mathrm{domain}(g)$ and all y, if $g(x) \mapsto y$, then $f(x) \mapsto y$. Let $\mathcal{F}$ and $\mathcal{G}$ be classes of partial multivalued functions. If f is a partial multivalued function, we define $f \in_c \mathcal{G}$ if $\mathcal{G}$ contains a refinement g of f, and we define $\mathcal{F} \subseteq_c \mathcal{G}$ if for every $f \in \mathcal{F}$, $f \in_c \mathcal{G}$.

Let *sat* be the multivalued function defined by $sat(x) \mapsto y$ if and only if x encodes a propositional formula and y encodes a satisfying assignment of x.

Let f and g be partial, multivalued functions. Then $g \leq_m^p f$ [FGH+96, Kre88] if there exist polynomial-time-computable total functions h and h' such that the partial, multivalued function defined by

$$g_1(x) = h'(x, f(h(x)))$$

is a refinement of g. A function $f \in \mathcal{F}$ is $\leq_m^p$*-complete* for the class $\mathcal{F}$ if for every $g \in \mathcal{F}$, $g \leq_m^p f$.

3 Existence of Complete Disjoint NP-Pairs

The following theorem is the main result of this section.

Theorem 3.1 *The following are equivalent.*

1. *There is a $\leq_m^{pp}$-complete disjoint* NP*-pair.*

2. *There is a smart $\leq_T^{pp}$-complete disjoint* NP*-pair.*

3. DisjNP *is uniformly enumerable.*

4. *There is a $\leq_{sm}^{pp}$-complete disjoint* NP*-pair.*

5. *There is a $\leq_{1\text{-}i}^{pp}$-complete disjoint* NP*-pair.*

6. $\mathrm{NPSV}_{\{0,1\}}$ *has a $\leq_m^p$-complete function.*

7. NPSV *has a $\leq_m^p$-complete function.*

Proof Köbler and Messner [KM00] have shown that items 4, 6, and 7 are equivalent. Therefore, it suffices to show that items 1 through 5 are equivalent.

Trivially, item 5 implies item 1 and item 1 implies item 2. To prove that item 2 implies item 3, let (A, B) be a smart $\leq_T^{pp}$-complete NP-pair. Assume that $A = L(N_A)$ and $B = L(N_B)$. In the following, we define a function f whose inputs are of the form $\langle i, j, k \rangle$. Given $\langle i, j, k \rangle$, define NP-machines N_1' and N_2' as follows.

- N_1' on input x simulates M_i on x. When M_i makes a query q, N_1' guesses a path of N_A on q, and a path of N_B on q. Since $A \cap B = \emptyset$, at most one of these paths will accept q. N_1' continues the simulation of M_i with a "yes" answer if the guessed path of N_A is an accepting path on q, and with a "no" answer if the guessed path of N_B is an accepting path of q. N_1' accepts x if and only the simulation of M_i successfully ends with acceptance of x and $x \in L(N_j)$.

- N_2' on input x simulates M_i on x identically to N_1'. However, N_2' accepts x if and only if the simulation of M_i successfully ends with rejection of x and $x \in L(N_k)$.

We note that for an arbitrary separator S of (A, B), $L(N_1')$ is not necessarily identical to $L(M_i^S)$: some $x \in L(M_i^S)$ belongs to $L(N_1')$ only if all the negative queries made by M_i on input x belong to B. Similarly, $L(N_2')$ is not necessarily $\overline{L(M_i^S)}$.

Let a and b be the indices of the NP machines N_1' and N_2' in the effective enumeration $\{N_i\}_i$. Define $f(\langle i,j,k\rangle) = (a,b)$. Note that f is total. Assume for some $\langle i,j,k\rangle$ that $f(\langle i,j,k\rangle) = (a,b)$ such that $L(N_a) \cap L(N_b) \neq \emptyset$. Then there exists some x such that the simulation of M_i on x has both a path where the simulation successfully ends with acceptance of x and a path where the simulation successfully ends with rejection of x. Hence, during the simulation there must be a query q such that $q \in L(N_a)$ and $q \in L(N_b)$. This cannot happen. Hence $L(N_a)$ and $L(N_b)$ are disjoint. So, for every i, j, and k, $(L(N_a), L(N_b)) \in \mathrm{DisjNP}$, where $(a,b) = f(\langle i,j,k\rangle)$.

Let $(C, D) \in \mathrm{DisjNP}$. Then $\exists j, k$ such that $C = L(N_j)$ and $D = L(N_k)$. Since (A, B) is smart $\leq_T^{pp}$-complete, there is some i such that (C, D) is smart reducible to (A, B) via M_i, i.e., if the input belongs to $C \cup D$, then the queries of M_i must belong to $A \cup B$. Let $f(\langle i,j,k\rangle) = (a,b)$. We claim that $C = L(N_a)$ and $D = L(N_b)$: If $x \in C$, then M_i on x accepts and every query is either in A or in B. Thus, the simulation of M_i on x by N_a will successfully end with acceptance of x. Since $x \in C = L(N_j)$, $x \in L(N_a)$ by definition. Conversely, if $x \in L(N_a)$, then $x \in L(N_j) = C$ by definition. So $C = L(N_a)$. Similarly, $D = L(N_b)$. This shows that DisjNP is uniformly enumerable.

Now we show that item 3 implies item 4. Assume that DisjNP is uniformly enumerable via some computable function f, and let M_f be a Turing machine that computes f. Let us define

$$A = \{0^n 1 0^t 1 x \mid M_f(n) = (i,j) \text{ within } t \text{ steps, and } N_i(x) \text{ accepts within } t \text{ steps}\} \tag{1}$$

and

$$B = \{0^n 1 0^t 1 x \mid M_f(n) = (i,j) \text{ within } t \text{ steps, and } N_j(x) \text{ accepts within } t \text{ steps}\}. \tag{2}$$

Then $A \in \mathrm{NP}$ and $B \in \mathrm{NP}$. If $A \cap B \neq \emptyset$, then $\exists n, t$ such that $0^n 1 0^t 1 x \in A \cap B$. So, $M_f(n) = (i,j)$ and $x \in L(N_i) \cap L(N_j)$. This is impossible because by item (1) of Definition 2.2, $L(N_i) \cap L(N_j) = \emptyset$. So $(A, B) \in \mathrm{DisjNP}$. We claim that (A, B) is $\leq_{sm}^{pp}$-complete.

Let $(C, D) \in \mathrm{DisjNP}$. For some n, $f(n) = (i,j)$, such that $C = L(N_i)$ and $D = L(N_j)$. Hence there exists l such that $M_f(n)$ outputs (i,j) within l steps. Let $p(\cdot)$ be the polynomial that bounds the running time of both N_i and N_j. Define $g(x) \stackrel{df}{=} 0^n 1 0^{\max\{l, p(|x|)\}} 1 x$. By the definition of A, $x \in C \Leftrightarrow g(x) \in A$; similarly, $x \in D \Leftrightarrow g(x) \in B$. Hence g is a $\leq_{sm}^{pp}$ reduction from (C, D) to (A, B).

Finally, we show that item 4 implies item 5. Let (A, B) be a $\leq_{sm}^{pp}$-complete pair. Define:

$$A' = \{\langle x, i, 0^t, a\rangle \mid f_i(x) = a \text{ within } t \text{ steps and } a \in A\} \tag{3}$$

and

$$B' = \{\langle x, i, 0^t, b\rangle \mid f_i(x) = b \text{ within } t \text{ steps and } b \in B\} \tag{4}$$

It is easy to see that (A', B') is in DisjNP; otherwise, if $\langle x, i, 0^t, w\rangle \in A' \cap B'$, then $w \in A \cap B$, contradicting $A \cap B = \emptyset$.

Let us assume that $(C, D) \in \mathrm{DisjNP}$. Therefore, $(C, D) \leq_{sm}^{pp} (A, B)$ via f_k for some k. Let us define a polynomial-time-computable total function

$$g(x) = \langle x, k, 0^{|x|^k + k}, f_k(x)\rangle.$$

We claim that $(C, D) \leq_{1\text{-}i}^{pp} (A', B')$ via g. If $x \in C$, then $f_k(x) \in A$, and therefore, $g(x) = \langle x, k, 0^{|x|^k+k}, f_k(x)\rangle \in A'$. Similarly, if $x \in D$, then $g(x) \in B'$. Clearly, g is an invertible reduction; $h(\langle x, k, 0^{|x|^k+k}, f_k(x)\rangle) = x$ satisfies that for every x, $h(g(x)) = x$. □

By Theorem 3.1, the following open questions are equivalent:

1. Does existence of a Turing-complete disjoint NP-pair imply existence of a many-complete disjoint NP-pair?
2. Does existence of a Turing-complete disjoint NP-pair imply existence of a smart Turing-complete disjoint NP-pair?

Note that if NPMV $\subseteq_c$ NPSV, then $sat \in_c$ NPSV. It is easy to see that this refinement of sat is $\leq_m^p$-complete for NPSV. Therefore, we obtain the following corollary of Theorem 3.1.

Corollary 3.2 *If* NPMV $\subseteq_c$ NPSV*, then* DisjNP *has a* $\leq_m^{pp}$*-complete pair.*

We do not expect that NPMV $\subseteq_c$ NPSV, because the assertion implies that the polynomial-time hierarchy collapses [HNOS96b]. We do not expect the assertions of Theorem 3.1 to be true either, but a proof would prove that NPMV $\subseteq_c$ NPSV is false also.

4 Smart Reductions

All the reductions in Theorem 3.1 are smart reductions. In the following theorem, we show under a reasonable complexity-theoretic hypothesis that there exist truth-table reductions that are not smart reductions.

Theorem 4.1 *If* UP $\cap$ co-UP *has a* P*-bi-immune set, then* DisjNP *contains disjoint pairs* (A, B) *and* (C, D) *such that* $(A, B) \leq_{tt}^{pp} (C, D)$ *but there is no smart reduction from* (A, B) *to* (C, D)*.*

Proof Let us define the following function:

$$dt(i) = \begin{cases} 1 & \text{if } i = 0 \\ 2^{2^{dt(i-1)}} & \text{otherwise.} \end{cases}$$

Let L be the set in $\mathrm{UP} \cap \text{co-UP}$ that is P-bi-immune. Consider

$$X = L \cap \{0^n \mid n = dt(i) \text{ for some } i\}.$$

We claim that X is infinite: Otherwise, if 0^l is the longest string in X, then $T_l = \{0^n \mid n > l \text{ and } n = dt(i) \text{ for some } i\}$ is an infinite subset of $\overline{L}$ that is in P. This contradicts P-bi-immunity of L. Similarly,

$$X' = \overline{L} \cap \{0^n \mid n = dt(i) \text{ for some } i\}$$

is also an infinite set. Both X and X' are in UP. Let us assume that $L(M) = X$, and $L(M') = X'$, where M and M' are UP machines, and the running time of both M and M' is bounded by some polynomial $p(\cdot)$.

We define the following machine N. If the input is not of the form $0^n, n = dt(i)$ for some i, then N rejects. Otherwise N guesses a bit. If the guessed bit is 0, N simulates M on 0^n, and accepts if and only if M accepts. If the guessed bit is 1, N simulates M' on 0^n, and accepts if and only if M' accepts. Every string of the form $0^n, n = dt(i)$ for some i, is either accepted by M or by M', but not by both machines. Therefore, N is a UP machine. Also, given an accepting computation of N, it is easy to determine whether the input belongs to $L(M)$ or to $L(M')$. Clearly, $L(N) = \{0^n \mid n = dt(i) \text{ for some } i\}$. For every such 0^n, let a_n be the accepting computation of N on 0^n.

Consider the following sets:

$$\begin{aligned}
A &= \{\langle 0^n, z\rangle \mid n = dt(i) \text{ for some } i \wedge z \leq a_n\}, \\
B &= \{\langle 0^n, z\rangle \mid n = dt(i) \text{ for some } i \wedge z > a_n\}, \\
C_1 &= \{\langle 0^n, k\rangle \mid n = dt(i) \text{ for some } i \wedge 0^n \in L(M) \wedge \text{ the } k\text{-th bit of the accepting} \\
&\quad\ \text{computation of } M \text{ on } 0^n \text{ is } 0\}, \\
D_1 &= \{\langle 0^n, k\rangle \mid n = dt(i) \text{ for some } i \wedge 0^n \in L(M) \wedge \text{ the } k\text{-th bit of the accepting} \\
&\quad\ \text{computation of } M \text{ on } 0^n \text{ is } 1\}, \\
C_2 &= \{\langle 1^n, k\rangle \mid n = dt(i) \text{ for some } i \wedge 0^n \in L(M') \wedge \text{ the } k\text{-th bit of the accepting} \\
&\quad\ \text{computation of } M' \text{ on } 0^n \text{ is } 0\}, \\
D_2 &= \{\langle 1^n, k\rangle \mid n = dt(i) \text{ for some } i \wedge 0^n \in L(M') \wedge \text{ the } k\text{-th bit of the accepting} \\
&\quad\ \text{computation of } M' \text{ on } 0^n \text{ is } 1\}.
\end{aligned}$$

Let us define

$$C = C_1 \cup C_2 \text{ and } D = D_1 \cup D_2.$$

It is easy to see that (A, B) and (C, D) are disjoint NP-pairs.

We show first that $(A, B) \leq_{tt}^{pp} (C, D)$. On input $\langle 0^n, z\rangle$, where $n = dt(i)$ for some i, the reduction machine asks for all possible bits of the accepting computations of M and M' on 0^n (i.e., the machine asks the following queries to (C, D): $\langle 0^n, 1\rangle, \ldots, \langle 0^n, p(n)\rangle$ and $\langle 1^n, 1\rangle, \ldots, \langle 1^n, p(n)\rangle$). Only one computation (of either M or M') is accepting. In polynomial time, the reduction machine can construct a_n, the accepting computation of N, and accepts if and only if $z \leq a_n$.

We now show that if $(A, B) \leq_T^{pp} (C, D)$ via a smart reduction, then $X \in \mathrm{P}$, contradicting the P-bi-immunity of L. Let M_S denote the machine that computes the smart reduction. Note that trivially $X \leq_T^{pp} (A, B)$; on input 0^n, where $n = dt(i)$ for some i, the reduction machine uses binary search to produce a_n, and accepts the input if and only if the first bit of a_n is 0. Let M_T denote the machine that computes this reduction.

To show that $X \in \mathrm{P}$, we will simulate M_T on input 0^n. If $n \neq dt(i)$ for some i, we reject 0^n. Otherwise, we will try to simulate the binary search algorithm of M_T. It is easy to see that if we can complete the binary search, then we can decide whether $0^n \in X$. However, it is possible that we may not be able to complete this binary search; in that case, we will show that we can accept or reject the input without obtaining a_n.

During simulation of M_T, when M_T makes a query $q = \langle 0^n, z\rangle$, we simulate the smart reduction machine M_S on q until M_S makes a query to (C, D). Since $n = dt(i)$, $0^n \in L(N)$ and therefore, a_n is defined and q belongs to $A \cup B$. Since M_S is a smart reduction machine, any query of M_S must belong to $C \cup D$. Let us assume that the first query that M_S makes is u.

We consider the following cases. If $u = \langle 0^n, k\rangle$, then $u \in C_1 \cup D_1$, and therefore, $0^n \in L(M)$, and therefore, $0^n \in X$. In this case, we accept the input and halt immediately. Similarly, if $u = \langle 1^n, k\rangle$, then $0^n \in L(M')$, and we halt and reject the input.

Assume that $u = \langle 0^m, k\rangle$, where $m \neq n$. We claim that $m < n$. Otherwise, since $u \in C \cup D$, $m = dt(j)$ for some $j > i$. However, in that case, $m \geq 2^{2^n}$, and M_T cannot write down u in polynomial time in n. Therefore, $m < n$. Again, this implies that $m = dt(j)$ for some $j < i$. Therefore, $n \geq 2^{2^m}$. In this case, we search for the accepting computation of M on 0^m in a brute-force manner. If there is an accepting computation, and the k-th bit of that computation is 0, the query is answered "yes"; otherwise, the query is answered "no". In either case, the query is answered correctly, and we continue the simulation of M_S on q. The case when $u = \langle 1^m, k\rangle$ is handled similarly; in this case, we search for an accepting computation of M'.

Since $m \leq \log\log n$, the brute-force search of the accepting computation of M or M' takes time $O(2^{p(m)})$, which is sublinear in n. Therefore, our simulation still takes polynomial time in n. We continue our simulation of $M_S(q)$, and each query is handled as above. If we do not accept or reject 0^n because of a halt, then we obtain correct answers to the queries, and at the end, we have answered the query q of M_T. In this way, we can continue the simulation. If the binary search is completed, we obtain the accepting path of N on 0^n, from which we can decide whether 0^n belongs to X. Note that in case of a halt we neither produce nor demand an accepting computation of N on 0^n.

Since our simulation takes polynomial-time in n, $X \in \mathrm{P}$. This completes the proof. □

5 Separation of many-one Reductions

Although existence of $\leq_m^{pp}$-complete pairs is equivalent to existence of $\leq_{sm}^{pp}$-complete and $\leq_{1\text{-}i}^{pp}$-complete pairs (Theorem 3.1), we show that these reductions are different. With trivial sets, this can be achieved easily. Consider $A = \{0\}$, $B = \{1\}$, $C = \{0\}$, and $D = \overline{C}$. Obviously

$(A, B) \leq_m^{pp} (C, D)$. However, $(A, B) \not\leq_{sm}^{pp} (C, D)$, since $\overline{C \cup D} = \emptyset$, and thus there is no space for strings in $\overline{A \cup B}$ to map to. Also, $(C, D) \leq_m^{pp} (A, B)$ but $(C, D) \not\leq_{1\text{-}i}^{pp} (A, B)$, since B is finite and D is infinite. Both these separations use finiteness in a crucial way. In the following, however, we achieve separations with infinite sets.

Theorem 5.1 *There exist disjoint NP-pairs (A, B) and (C, D) such that A, B, C, D, $\overline{A \cup B}$, and $\overline{C \cup D}$ are infinite, and $(A, B) \leq_m^{pp} (C, D)$ but $(A, B) \not\leq_{1\text{-}i}^{pp} (C, D)$.*

Proof Let us define the following sets:

$$\begin{aligned} A &\overset{df}{=} \{00x \mid x \in \Sigma^*\}, \\ B &\overset{df}{=} \{11x \mid x \in \Sigma^*\}, \\ C &\overset{df}{=} \{0^n \mid n \geq 0\}, \\ D &\overset{df}{=} \{1^n \mid n \geq 0\}. \end{aligned}$$

Clearly, $(A, B) \leq_m^{pp} (C, D)$ via $f(x) = 0^{|x|}$ if $x \in A$, and $f(x) = 1^{|x|}$ if $x \in B$, and $f(x) = 01$ otherwise. (Note that f is actually a $\leq_{sm}^{pp}$-reduction.) We claim that $(A, B) \not\leq_{1\text{-}i}^{pp} (C, D)$ via any polynomial-time-computable total function g. Otherwise, let g be a function that is computable in time n^k. Then, any string of length n in A can be mapped to a string of length at most n^k in C. There are $n^k + 1$ strings in C of length at most n^k, but there are 2^{n-2} strings of length n in A. Therefore, g cannot be one-to-one, and hence, cannot be inverted. □

Theorem 5.2 *The following are equivalent:*

1. P $\neq$ NP.
2. *There are disjoint* NP*-pairs (A, B) and (C, D) such that A, B, C, D, $\overline{A \cup B}$, and $\overline{C \cup D}$ are infinite, and $(A, B) \leq_m^{pp} (C, D)$ but $(A, B) \not\leq_{sm}^{pp} (C, D)$.*

Proof If $\mathrm{P} = \mathrm{NP}$, then given disjoint NP-pairs (A, B) and (C, D), A, B, C, and D are all in P. Given any string x, it can be determined whether $x \in A$, $x \in B$, or $x \in \overline{A \cup B}$, and x can be mapped appropriately to some fixed string in C, D, or $\overline{C \cup D}$. Therefore, $(A, B) \leq_{sm}^{pp} (C, D)$.

For the other direction, consider the clique-coloring pair. This is a disjoint NP-pair, and is known to be P-separable [Lov79, Pud01]:

$$C_1 = \{\langle G, k\rangle \mid G \text{ has a clique of size } k\}, \tag{5}$$

and

$$C_2 = \{\langle G, k\rangle \mid G \text{ has a coloring with } k-1 \text{ colors}\}. \tag{6}$$

Let S be the separator that is in P. Note that $(C_1, C_2) \leq_m^{pp} (S, \overline{S})$ via the identity function. (Note that this reduction is also invertible.) Let

$$C = \{\langle G, 3\rangle \mid G \text{ is a cycle of odd length with at least 5 vertices}\}.$$

Let $S_1 = S - C$ and $S_2 = S - C$. Both S_1 and S_2 are in P. Since any odd cycle with at least 5 vertices is not 2-colorable, and does not contain any clique of size 3, $C \cap C_1 = \emptyset$, and $C \cap C_2 = \emptyset$. Therefore, $(C_1, C_2) \leq_m^{pp} (S_1, S_2)$ via the identity function. Assume that $(C_1, C_2) \leq_{sm}^{pp} (S_1, S_2)$. Then $C_1 \leq_m^p S_1$, and $C_2 \leq_m^p S_2$. Hence C_1 and C_2 are in P. This is impossible, since $\mathrm{NP} \neq \mathrm{P}$, and C_1 and C_2 are NP-complete. Thus, $(C_1, C_2) \not\leq_{sm}^{pp} (S_1, S_2)$. □

6 Separating Adaptive and Nonadaptive Reductions

Glaßer et. al [GSSZ03] provided evidence showing that $\leq_m^{pp}$ reductions between disjoint NP-pairs are not the same as $\leq_{1\text{-}tt}^{pp}$ reductions between disjoint NP-pairs. In the following theorems, we separate $\leq_{btt}^{pp}$ from $\leq_{tt}^{pp}$ and $\leq_{tt}^{pp}$ from $\leq_T^{pp}$ using reasonable complexity-theoretic hypotheses. Our separations are obtained easily from existing techniques to separate reductions between NP sets [PS01].

A set L is *p-selective* if there is a polynomial-time-bounded function g such that for every $x, y \in \Sigma^*$, $g(x,y) \in \{x,y\}$, and $\{x,y\} \cap L \neq \emptyset \Rightarrow g(x,y) \in L$ [Sel79]. The function g is called the *selector function* for L.

Given a finite alphabet, let Σ^ω denote the set of all strings of infinite length of order type ω. For $r \in \Sigma^* \cup \Sigma^\omega$, the standard left cut of r [Sel79, Sel82] is the set

$$L(r) = \{x \,\big|\, x < r\},$$

where $<$ is the ordinary dictionary ordering of strings with 0 less than 1. It is obvious that every standard left cut is p-selective with selector $g(x,y) = \min(x,y)$.

For any $A \in \mathrm{NP}$, there is a polynomial $p(\cdot)$, and a polynomial-time predicate R such that

$$x \in A \Leftrightarrow \exists y[|y| \leq p(|x|) \wedge R(x,y)].$$

We say that R and p define A, and a string y that satisfies the above equation is called a *witness* for x. For any $A \in \mathrm{NP}$, and R and p that define A, we define the partial multivalued function $f_{R,p}$ that maps input strings to witnesses as follows:

$$f_{R,p}(x) \mapsto y, \text{ if } |y| \leq p(|x|) \text{ and } R(x,y).$$

Definition 6.1 ([HNOS96a]) *If* $f_{R,p} \leq_{tt}^p A$*, then* search nonadaptively reduces to decision *for* A.

Hemaspaandra et. al [HNOS96a] credit Thierauf for the following proposition.

Proposition 6.2 (Thierauf [HNOS96a]) *If* $L \in \mathrm{NP}$ *is* $\leq_{tt}^p$*-reducible to a p-selective set and search nonadaptively reduces to decision for* L*, then* $L \in \mathrm{P}$.

We also need the following easy proposition.

Proposition 6.3 *For* $\mu \in \{m, btt, tt, T\}$*, it holds that* $(A, \overline{A}) \leq_\mu^{pp} (B, \overline{B})$ *if and only if* $A \leq_\mu^p B$.

Theorem 6.4 *If* $\mathrm{UE} \cap \text{co-UE} \neq \mathrm{E}$*, then there are pairs* (A,B) *and* (C,D) *in* DisjNP *such that* $(A,B) \leq_{tt}^{pp} (C,D)$*, but* $(A,B) \not\leq_{btt}^{pp} (C,D)$.

Proof Since $\mathrm{UE} \cap \text{co-UE} \neq \mathrm{E}$, there must be a tally set $T \in (\mathrm{UP} \cap \text{co-UP}) - \mathrm{P}$. Let R and R' be the polynomial-time-decidable predicates associated with T and $\overline{T}$ respectively. We define the following languages:

$$L_1 = \{(0^n, z) \,\big|\, \exists y R(0^n, y) \text{ and } z \leq y\}, \tag{7}$$

and

$$L_2 = \{(0^n, i) \,\big|\, \exists y R(0^n, y) \text{ and } i\text{-th bit of } y \text{ is } 1\}. \tag{8}$$

It is easy to see that both L_1 and L_2 are in UP. To see that they are also in co-UP, note that

$$\overline{L_1} = \{(0^n, z) \,\big|\, (\exists y R'(0^n, y)) \text{ or } (\exists y R(0^n, y) \text{ and } z > y)\}.$$

For any 0^n, either there exists y such that $R(0^n, y)$ holds, or there exists y such that $R'(0^n, y)$ holds, but both cannot hold simultaneously. Therefore, $\overline{L_1}$ belongs to UP. Similarly, since

$$\overline{L_2} = \{(0^n, i) \,\big|\, (\exists y R'(0^n, y)) \text{ or } (\exists y R(0^n, y) \text{ and } i\text{-th bit of } y \text{ is } 0)\},$$

L_2 is also in co-UP. Therefore, $(L_1, \overline{L_1})$, and $(L_2, \overline{L_2})$ are both in DisjNP.

It is clear that $L_1 \leq^p_{tt} L_2$. Observe that L_2 is a sparse set. Ogihara and Watanabe [OW91] call L_1 the *left set* of T, and they and Homer and Longpré [HL94] proved for every T in NP that if the left set of T is $\leq^p_{btt}$-reducible to a sparse set, then T is in P. Therefore, $L_1 \not\leq^p_{btt} L_2$.

By Proposition 6.3, we have that $(L_1, \overline{L_1}) \leq^{pp}_{tt} (L_2, \overline{L_2})$, but $(L_1, \overline{L_1}) \not\leq^{pp}_{btt} (L_2, \overline{L_2})$. □

Theorem 6.5 *If* $\text{UE} \cap \text{co-UE} \neq \text{E}$*, then there are pairs* (A, B) *and* (C, D) *in* DisjNP *such that* $(A, B) \leq^{pp}_T (C, D)$*, but* $(A, B) \not\leq^{pp}_{tt} (C, D)$.

Proof Since $\text{UE} \cap \text{co-UE} \neq \text{E}$, there must be a tally set $T \in (\text{UP} \cap \text{co-UP}) - \text{P}$. Let us assume that $T = L(M)$, and $\overline{T} = L(M')$, where both M and M' are UP machines. For every n, 0^n is either accepted by M or by M', but not by both. Let a_n be the accepting computation of M or M' on 0^n. Note that this is well-defined. We define the following infinite string $a = a_1 a_2 \cdots$, and let

$$L(a) = \{x \,\big|\, x < a\}$$

be the standard left cut of a. Note that $L(a) \in \text{UP} \cap \text{co-UP}$ and is p-selective. We define

$$L = \{0^{\langle n,i \rangle} \,\big|\, \exists y, y = a_n \text{ and } i\text{-th bit of } y \text{ is } 0 \,\}.$$

Note that $L \in \text{UP} \cap \text{co-UP}$. Also observe that $L \notin \text{P}$; otherwise, $T \in \text{P}$ as well, contradicting our assumption.

It is easy to see that $L \leq^p_T L(a)$: On input $0^{\langle n,i \rangle}$, the reduction machine can use binary search with $L(a)$ as the oracle and can determine a_n, and accept the input if and only if the i-th bit of a_n is 0.

We claim that $L \not\leq^p_{tt} L(a)$. It is clear that *search nonadaptively reduces to decision* for L, since on input $0^{\langle n,i \rangle}$, a_n can be obtained by nonadaptive queries to L. Then by Proposition 6.2, $L \leq^p_{tt} L(a)$ would imply that $L \in \text{P}$, which is a contradiction. By Proposition 6.3, we have that $(L, \overline{L}) \leq^{pp}_T (L(a), \overline{L(a)})$, but $(L, \overline{L}) \not\leq^{pp}_{tt} (L(a), \overline{L(a)})$. □

7 Oracle Construction

To study further the open question of whether existence of a Turing-complete disjoint NP-pair implies existence of a many-one-complete disjoint NP-pair, in this section we construct an oracle relative to which DisjNP has a truth-table-complete disjoint NP-pair, but does not have any many-one-complete disjoint NP-pair. Hemaspaandra *et al.* [HHH98] asked whether there are natural complexity classes for which the existence of many-one and Turing-complete sets can be distinguished, that is, classes that in some relativized world simultaneously have Turing-complete sets and lack many-one-complete sets. By Theorem 7.3 below, DisjNP is such a class.

Define $\mathrm{UP} \vee \mathrm{UP} \overset{df}{=} \{L_0 \cup L_1 \,|\, L_0, L_1 \in \mathrm{UP}\}$. For a finite $Y \subseteq \Sigma^*$, let $\ell(Y) \overset{df}{=} \sum_{w \in Y} |w|$. For a path P of some nondeterministic computation, P^{yes} (resp., P^{no}) denotes the set of oracle queries that are answered positively (resp., negatively) along P. Let $|P|$ denote the length of P.

Theorem 7.1 *If* $\mathrm{NP} = \mathrm{UP} \vee \mathrm{UP}$*, then there exists a disjoint* NP*-pair that is* $\leq_{tt}^{pp}$*-hard for* NP.

Proof By assumption, $\mathrm{SAT} = L_0 \cup L_1$ for $L_0, L_1 \in \mathrm{UP}$. Let M_0 and M_1 be UP-machines for L_0 and L_1. Let $L \overset{df}{=} 0L_0 \cup 1L_1$. Note that $L \in \mathrm{UP}$ via the following UP-machine M: On input x, M extracts the first bit b from x. The remaining string is denoted by x'. If $b = 0$, then M simulates $M_0(x')$. Otherwise M simulates $M_1(x')$. Let p denote the running time of M.

$$\begin{aligned} A_0 &\overset{df}{=} \{0^k 1w \,|\, w \in L \text{ and the } k\text{-th bit of the accepting path of } M(w) \text{ is } 0\} \\ A_1 &\overset{df}{=} \{0^k 1w \,|\, w \in L \text{ and the } k\text{-th bit of the accepting path of } M(w) \text{ is } 1\} \end{aligned}$$

Observe that $(A_0, A_1) \in \mathrm{DisjNP}$. We show $\mathrm{SAT} \leq_{tt}^{pp} (A_0, A_1)$ via the following reduction: On input x, the machine asks all queries $0^k 10x$ and $0^k 11x$ for $1 \leq k \leq p(|x|)$. Let a_0 and a_1 denote the corresponding vectors of answers. The reduction machine accepts if either a_0 is an accepting path of $M_0(x)$, or a_1 is an accepting path of $M_1(x)$.

If the reduction machine accepts x, then either $x \in L_0$ or $x \in L_1$, and therefore, $x \in \mathrm{SAT}$. On the other hand, if $x \in \mathrm{SAT}$, then x is either in L_0 or in L_1. Without loss of generality assume $x \in L_0$. It follows that $0x \in L$ and therefore, a_0 gives us the accepting path of $M(0x)$. By construction of M, this is also the accepting path of $M_0(x)$. Therefore, the reduction machine accepts. $\square$

Corollary 7.2 *If* $\mathrm{NP} = \mathrm{UP} \vee \mathrm{UP}$*, then* DisjNP *has* $\leq_{tt}^{pp}$*-complete pairs.*

Even *et al.* [ESY84] conjectured that there do not exist disjoint NP-pairs that are $\leq_T^{pp}$-hard for NP. Therefore, if $\mathrm{NP} = \mathrm{UP} \vee \mathrm{UP}$, then this conjecture does not hold.

Theorem 7.3 *There exists an oracle X relative to which* DisjNP *has* $\leq_{tt}^{pp}$*-complete pairs, but no* $\leq_m^{pp}$*-complete pairs.*

Proof We construct the oracle such that $\mathrm{NP}^X = \mathrm{UP}^X \vee \mathrm{UP}^X$ and there do not exist $\leq_m^{pp,X}$-complete disjoint NP^X-pairs. Define

$$\begin{aligned} A_{i,j} &\overset{df}{=} \{0^n \,|\, \exists y \text{ such that } |000^i 10^j 1y| = n \text{ and } 000^i 10^j 1y \in X\} \\ B_{i,j} &\overset{df}{=} \{0^n \,|\, \exists y \text{ such that } |010^i 10^j 1y| = n \text{ and } 010^i 10^j 1y \in X\} \end{aligned}$$

Note that $A_{i,j}$ and $B_{i,j}$ depend only on oracle words that start with letter 0. We will seek either to make the pair $(L(M_i^X), L(M_j^X))$ not disjoint (in this case $A_{i,j} \cap B_{i,j}$ may not be empty), or to show that $(L(M_i^X), L(M_j^X))$ is not a many-one complete pair (in this case $(A_{i,j}, B_{i,j})$ will be a disjoint NP^X-pair). Define the canonical NP^X-complete set as

$$C = \{\langle 0^n, 0^t, x\rangle \mid M_n^X(x) \text{ accepts within } t \text{ steps}\}.$$

We construct X such that it satisfies two conditions.

C1: $\langle 0^n, 0^t, x\rangle \in C \Leftrightarrow \exists y_0, |y_0| = |100^n10^t1x|[100^n10^t1xy_0 \in X]$ or
$\exists y_1, |y_1| = |110^n10^t1x|[110^n10^t1xy_1 \in X]$

C2: $\forall n, t, x$ there exists at most one y_0 and at most one y_1

These two conditions describe the coding part of the oracle X. Words of the forms $100^n10^t1xy_0$ and $110^n10^t1xy_1$ are called *codewords*. Codewords always start with 1. Since these codewords correspond to the computation of $M_n(x)$ restricted to t steps, we call $M_n(x)$ also the computation that corresponds to these codewords. If we say that C1 or C2 hold for a finite oracle $Z' \subseteq \Sigma^{\leq m}$, then we mean that these conditions (this time with Z' instead of X) hold for all words up to length m.

If both C1 and C2 hold, then $\mathrm{NP}^X = \mathrm{UP}^X \vee \mathrm{UP}^X$. In the remaining proof we show that we can diagonalize against every potential $\leq_m^{pp,X}$-complete pair $(L(M_i^X), L(M_j^X))$ and every possible reduction function f while maintaining C1 and C2. This shows that $\leq_m^{pp,X}$-complete disjoint NP^X-pairs do not exist, yet $\mathrm{NP}^X = \mathrm{UP}^X \vee \mathrm{UP}^X$. From Corollary 7.2 it follows that there exist $\leq_{tt}^{pp,X}$-complete disjoint NP^X-pairs.

Let Z be the finite oracle constructed so far, say up to words of length $\leq k-1$. Our construction ensures that k is large enough such that the membership of words of length $\geq k$ does not affect diagonalizations made in previous steps. Let i and j be given indices of nondeterministic polynomial-time oracle Turing machines, and let f be a given polynomial-time oracle function. Assume that the running time of $f(x)$, $M_i(x)$, $M_j(x)$, $M_i(f(x))$, and $M_j(f(x))$ is bounded by the polynomial r (independent of the oracle). Starting from Z we construct a finite extension Z' that forces that either

$$L(M_i^X) \cap L(M_j^X) \neq \emptyset, \tag{9}$$

or

$$(A_{i,j}, B_{i,j}) \not\leq_m^{pp,X} (L(M_i^X), L(M_j^X)) \text{ via reduction function } f^X. \tag{10}$$

We can assume that k is large enough such that $(5 \cdot r(k))^2 \leq 2^{k/2}$. Otherwise we continue the construction while doing coding for C1 and C2 until we reach a stage k that is large enough.

We define the notion of reservations for computations. A reservation consists of disjoint sets Y and N where Y contains words that are reserved for the oracle (i.e., yes answers) while N contains words that are reserved for the complement of the oracle (i.e., no answers).

Call a pair (Y, N) a *reservation* if Y and N are subsets of $\Sigma^{\geq k}$, $Y \cap N = \emptyset$, $\ell(Y \cup N) \leq 5 \cdot r(k)$, condition C2 holds for Y, and if $w \in Y$ is a codeword for some computation $M_n(x)$, then $M_n^{Z\cup Y}(x)$ has an accepting path P such that $P^{\text{yes}} \cap \Sigma^{\geq k} \subseteq Y$ and $P^{\text{no}} \cap \Sigma^{\geq k} \subseteq N$.

Claim 7.4 *For every reservation (Y, N) there exists an extension Z' of Z such that Z' is defined up to length $r(k)$, Z' satisfies C1 and C2, $Y \subseteq Z'$ and $N \subseteq \overline{Z'}$.*

Proof The extension Z' is constructed as follows. We start with oracle Z and add codewords in order to achieve C1. If a codeword with prefix 100^n10^t1x or 110^n10^t1x needs to be added to Z', and if a word with such a prefix is already in Y, then we add that codeword. Otherwise, we choose an appropriate codeword that is not in N. This can be done since for any length $l \geq k$, the number of possible y_0 and y_1 (as required by C1) is $2^{l/2} \geq 2^{k/2}$, while $\|N\| \leq 5 \cdot r(k)$. Moreover, in our construction, we add all words from Y to the oracle. This is possible since by definition of reservations, whenever some w is in Y, the computation corresponding to w is forced to accept (since we fixed the queries of an accepting path). Therefore, we can add every $w \in Y$ to the oracle without violating C1. Finally, Z' satisfies C2, since Y does so and we add at most one codeword for every 100^n10^t1x and for every 110^n10^t1x. □

Let N_f be the set of words in $\Sigma^{\geq k}$ that are queried by the computation $f(0^k)$ using oracle Z. Words in N_f are reserved for the complement of X. We restrict the notion of reservations as follows. Call a reservation (Y, N) a *reservation for* $M_i(f(0^k))$ if $\ell(Y \cup N) \leq 2 \cdot r(k)$, $Y \cap N_f = \emptyset$, all codewords in Y start with 10, and $M_i^{Z \cup Y}(f(0^k))$ has an accepting path P such that $P^{\text{yes}} \cap \Sigma^{\geq k} \subseteq Y$ and $P^{\text{no}} \cap \Sigma^{\geq k} \subseteq N$. Analogously we define *reservations for* $M_j(f(0^k))$; here all codewords in Y have to start with 11, and Y (resp., N) contains positive (resp., negative) queries made on some accepting path of $M_j^{Z \cup Y}(f(0^k))$.

Claim 7.5 *Let Z' be an extension of Z such that $Z' \cap N_f = \emptyset$ and Z' is defined up to words of length $\leq r(k)$. If Z' satisfies C1 and C2, all codewords in ${Z'}^{\geq k}$ start with* 10, *and $M_i^{Z'}(f(0^k))$ accepts, then there exists a reservation (Y', N') for $M_i(f(0^k))$ such that $Y' \subseteq Z'$ and $N' \subseteq \overline{Z'}$.*

The analogous claim holds for codewords starting with 11 and for computation $M_j(f(0^k))$.

Proof For every $Y \subseteq {Z'}^{\geq k}$ define the set of dependencies as

$$\mathcal{D}(Y) \overset{df}{=} \{q \mid Y \text{ contains a codeword that corresponds to the computation } M_n^{Z'}(x) \text{ restricted to } t \text{ steps and } q \in P_{n,t,x}^{\text{all}}\},$$

where $P_{n,t,x}$ is the lexicographically smallest path among all paths of $M_n^{Z'}(x)$ that are accepting and that are of length $\leq t$. The path $P_{n,t,x}$ exists, since C1 holds for Z'.

If w is a codeword for the computation $M_n(x)$ restricted to t steps, then $|P_{n,t,x}| \leq t < |w|/2$. Therefore, the sum of lengths of q's that are induced by some codeword w in Y is at most $|w|/2$. This shows for all $Y \subseteq {Z'}^{\geq k}$ that

$$\ell(\mathcal{D}(Y)) \leq \ell(Y)/2. \tag{11}$$

Let P be an accepting path of $M_i^{Z'}(f(0^k))$. The procedure below computes the reservation (Y', N') for $M_i(f(0^k))$.

```
1    $Y' := P^{yes} \cap \Sigma^{\geq k}$
2    $N' := P^{no} \cap \Sigma^{\geq k}$
3    $c := 0$
4    repeat
5        $c := c + 1$
```

```
6            $Y_c := \mathcal{D}(Y_{c-1}) \cap Z'^{\geq k}$
7            $N_c := \mathcal{D}(Y_{c-1}) \cap \overline{Z'}^{\geq k}$
8            $Y' = Y' \cup Y_c$
9            $N' = N' \cup N_c$
10   until $Y_c = N_c = \emptyset$
```

Clearly, $Y' \subseteq Z'$ and $N' \subseteq \overline{Z'}$. Therefore, $Y' \cap N' = \emptyset$. By lines 6 and 7, and by Equation (11), the following holds for $1 \leq i \leq c$.

$$\ell(Y_i \cup N_i) \leq \ell(\mathcal{D}(Y_{i-1})) \leq \ell(Y_{i-1})/2$$

Hence the procedure terminates and

$$\ell(Y' \cup N') \leq 2 \cdot \ell(Y_0 \cup N_0) \leq 2 \cdot r(k).$$

Condition C2 holds for Y', since it holds for Z'. Assume $w \in Y'$ is a codeword for some computation $M_n(x)$ restricted to t steps. Hence $w \in Y_c$ for some c. $M_n^{Z'}(x)$ accepts within t steps, since C1 holds for Z'. Therefore, $P^{\text{all}}_{n,t,x} \subseteq \mathcal{D}(Y_c)$. It follows that

$$P^{\text{yes}}_{n,t,x} \cap \Sigma^{\geq k} \subseteq Y_{c+1} \subseteq Y'$$

and

$$P^{\text{no}}_{n,t,x} \cap \Sigma^{\geq k} \subseteq N_{c+1} \subseteq N'.$$

This shows that (Y', N') is a reservation.

It remains to show that (Y', N') is a reservation for $M_i(f(0^k))$. Since $Y' \subseteq Z'^{\geq k}$, $Y' \cap N_f = \emptyset$ and all codewords in Y' start with 10. Since $Y_0 \subseteq Y'$ and $N_0 \subseteq N'$, P is an accepting path of $M_i^{Z \cup Y'}(f(0^k))$ such that $P^{\text{yes}} \cap \Sigma^{\geq k} \subseteq Y'$ and $P^{\text{no}} \cap \Sigma^{\geq k} \subseteq N'$. □

We define sets of reservations.

- R_0 is the set of all reservations for $M_i(f(0^k))$.
- R_1 is the set of all reservations for $M_j(f(0^k))$.

Every codeword in a reservation that belongs to R_0 starts with 10, and every codeword in a reservation that belongs to R_1 start with 11. If we could do the construction using only one type of reservation (either those in R_0 or those in R_1), then this would give NP $=$ UP. However, we will see that sometimes we have to combine a reservation from R_0 with a reservation from R_1. For this reason we obtain only NP $=$ UP $\vee$ UP.

We say that a reservation $(Y_0, N_0) \in R_0$ *conflicts* with a reservation $(Y_1, N_1) \in R_1$ if either $Y_0 \cap N_1 \neq \emptyset$ or $Y_1 \cap N_0 \neq \emptyset$.

Assume that there exist $(Y_0, N_0) \in R_0$ and $(Y_1, N_1) \in R_1$ that do not conflict. Let $Y = Y_0 \cup Y_1$ and $N = N_0 \cup N_1 \cup N_f$. Observe that (Y, N) is a reservation. By Claim 7.4, there exists an extension Z' of Z such that Z' is defined up to length $r(k)$, Z' satisfies C1 and C2, $Y \subseteq Z'$ and $N \subseteq \overline{Z'}$. This ensures that both $M_i^{Z'}(f(0^k))$ and $M_j^{Z'}(f(0^k))$ accept. Therefore, $(L(M_i^X), L(M_j^X))$ is not in DisjNP^X, and Equation (9) holds. So in this case we have successfully diagonalized against the pair $(L(M_i^X), L(M_j^X))$, and we can proceed to the next stage of the construction.

For the rest of the proof, we assume that every reservation in R_0 conflicts with every reservation in R_1. We will prove under this assumption that Equation (10) holds. The idea is as follows. In Claim 7.6, we construct a small set of words N such that any extension Z' of Z that does not contain any word in $N \cup N_f$ will force either $M_i^{Z'}(f(0^k))$ or $M_j^{Z'}(f(0^k))$ to reject. Putting an appropriate word of the form 000^i10^j1y (resp., 010^i10^j1y) in Z' will ensure that 0^k is in $A_{i,j}$ (resp., in $B_{i,j}$), thereby ensuring that Equation (10) is true. The details follow.

Assumption: *Every reservation in R_0 conflicts with every reservation in R_1.*

Claim 7.6 *There exists an $N \subseteq \Sigma^{\leq r(k)}$ such that $\|N\| \leq (2 \cdot r(k))^2$ and*

- *either for all $(Y_0, N_0) \in R_0$, $Y_0 \cap N \neq \emptyset$*
- *or for all $(Y_1, N_1) \in R_1$, $Y_1 \cap N \neq \emptyset$.*

Proof We create N as follows.

```
1    $N = \emptyset$
2    while ($R_0 \neq \emptyset$ and $R_1 \neq \emptyset$)
3        Choose some $(Y^*, N^*) \in R_0$
4        $N = N \cup Y^* \cup N^*$
5        For every $(Y_0, N_0) \in R_0$
6           if $Y_0 \cap (Y^* \cup N^*) \neq \emptyset$ then remove $(Y_0, N_0)$
7        For every $(Y_1, N_1) \in R_1$
8           if $Y_1 \cap (Y^* \cup N^*) \neq \emptyset$ then remove $(Y_1, N_1)$
9    end while
```

We claim that after n iterations of the *while* loop, for every $(Y_1, N_1) \in R_1$, $\|N_1\| \geq n$. If this is true, then the *while* loop iterates at most $2 \cdot r(k)$ times, since for any $(Y_1, N_1) \in R_1$, $\|N_1\| \leq 2 \cdot r(k)$. On the other hand, during each iteration, N is increased by at most $2 \cdot r(k)$ strings, since for any $(Y_0, N_0) \in R_0$, $\|Y_0 \cup N_0\| \leq 2 \cdot r(k)$. Therefore, when the algorithm terminates, $\|N\| \leq (2 \cdot r(k))^2$. Also, if R_0 is empty, then for every (Y_0, N_0) that has been removed from R_0, $Y_0 \cap N \neq \emptyset$; and if R_1 is empty, then for every (Y_1, N_1) that has been removed from R_1, $Y_1 \cap N \neq \emptyset$.

It remains to prove that after the n-th iteration of the *while* loop, for every $(Y_1, N_1) \in R_1$, $\|N_1\| \geq n$.

For every n, let (Y^n, N^n) be the reservation that is chosen during the n-th iteration in step 3. For every (Y_1, N_1) that is in R_1 at the beginning of this iteration, (Y^n, N^n) conflicts with (Y_1, N_1) (by assumption). Therefore, there is a word in $(N^n \cap Y_1) \cup (Y^n \cap N_1)$. If this word is in $N^n \cap Y_1$, then (Y_1, N_1) will be removed from R_1 in step 8. Otherwise, i.e., if $Y^n \cap N_1 \neq \emptyset$, then let w be the lexicographically smallest word in $Y^n \cap N_1$. In this case, (Y_1, N_1) will not be removed from R_1. We say that (Y_1, N_1) *survives* the n-th iteration *due to* w. Note that (Y_1, N_1) can survive only due to a word that is in N_1. We will use this fact to prove that $\|N_1\| \geq n$ after n iterations.

We show that any reservation that is left in R_1 after n iterations survives each iteration due to a different word. Assume that (Y_1, N_1) survives iteration n due to $w \in Y^n \cap N_1$. If (Y_1, N_1) had survived an earlier iteration $l < n$ due to the same word, then w is also in $Y^l \cap N_1$. Therefore, $Y^l \cap Y^n \neq \emptyset$. So (Y^n, N^n) should have been removed in step 6 during iteration l, and cannot be chosen at the beginning of iteration n. Hence, w cannot be the query by which (Y_1, N_1) had survived iteration l. □

Let N be as in Claim 7.6. Without loss of generality, we assume that for all $(Y_0, N_0) \in R_0$, $Y_0 \cap N \neq \emptyset$. Add all words from N_f to N. Now $\|N\| \leq (3 \cdot r(k))^2$. We consider the words in N to be reserved for the complement of X.

Claim 7.7 *Let Z' be any extension of Z such that Z' is defined up to length $r(k)$. If Z' satisfies C1 and C2, all codewords in $Z'^{\geq k}$ start with 10, and $Z' \cap N = \emptyset$, then $M_i^{Z'}(f(0^k))$ rejects.*

The analogous claim holds for codewords starting with 11 and for computation $M_j^{Z'}(f(0^k))$.

Proof Assume that $M_i^{Z'}(f(0^k))$ accepts. Note that $Z' \cap N_f = \emptyset$. By Claim 7.5, there exists a reservation (Y', N') for $M_i(f(0^k))$ such that $Y' \subseteq Z'$ and $N' \subseteq \overline{Z'}$. By definition, (Y', N') belongs to R_0. Therefore, by assumption, $Y' \cap N \neq \emptyset$. Hence $Z' \cap N \neq \emptyset$, a contradiction. □

Choose a word $w \in \Sigma^k - N$ that is of the form $w = 000^i10^j1y$. Add w to the oracle Z. We continue the construction by making only coding for C1 and C2. For this we use only codewords that start with 10 while we reserve words in N for the complement of the oracle. This is possible since the number of words in N is small. Let Z' be the resulting oracle that is now defined up to oracle stage $r(k)$. Note that $0^k \in A_{i,j}$ is witnessed by $w \in Z' \subseteq X$. By Claim 7.7, $M_i^{Z'}(f(0^k))$ rejects. This computation cannot ask queries longer than $r(k)$: For any X that is an extension of Z', $M_i^X(f(0^k))$ rejects as well. Therefore, relative to X, $(A_{i,j}, B_{i,j})$ does not $\leq_m^{pp}$-reduce to $(L(M_i^X), L(M_j^X))$ via reduction function f. This completes the proof of Theorem 7.3. □

References

[BLS84] R. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM Journal on Computing*, 13(3):461–487, 1984.

[ESY84] S. Even, A. Selman, and J. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61:159–173, 1984.

[FGH$^+$96] S. Fenner, F. Green, S. Homer, A. Selman, T. Thierauf, and H. Vollmer. Complements of multivalued functions. In *Proceedings 11th Conference on Computational Complexity*, pages 260–269. IEEE Computer Society Press, 1996.

[GG98] O. Goldreich and S. Goldwasser. The possibility of basing cryptography on the assumption that P $\neq$ NP. Available from *http://theory.lcs.mit.edu/~tcryptol*, 1998.

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

[GSSZ03] C. Glaßer, A. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. In *Proceedings 18th Computational Complexity*. IEEE Computer Society, 2003.

[HHH98] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel. $R^{SN}_{1-tt}(\mathrm{NP})$ distinguishes robust many-one and Turing completeness. *Theory of Computing Systems*, 31(3):307–325, 1998.

[HL94] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. *Journal of Computer and Systems Sciences*, 48(2):324–336, 1994.

[HNOS96a] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *Journal of Computer and System Sciences*, 53:194–209, 1996. Special Issue of papers selected from the Eighth Annual IEEE Conference on Structure in Complexity Theory.

[HNOS96b] L. Hemaspaandra, A. Naik, M. Ogihara, and A. Selman. Computing solutions uniquely collapses the polynomial hierarchy. *SIAM Journal on Computing*, 25:697–708, 1996.

[KM00] J. Köbler and J. Messner. Is the standard proof system for sat p-optimal? In *Proceedings of the 20th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1974 of *Lecture Notes in Computer Science*, pages 361–372. Springer Verlag, 2000.

[KMT03] J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 2003. To appear.

[Kre88] M. W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[Lov79] L. Lovász. On the shannon capacity of graphs. *IEEE Transactions on Information Theory*, 25:1–7, 1979.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing*, 20(3):471–483, 1991.

[PS01] A. Pavan and A. Selman. Separation of NP-completeness notions. In *Proceedings 16th IEEE Conference on Computational Complexity*. IEEE Computer Society, 2001.

[Pud01] P. Pudlák. On reducibility and symmetry of disjoint NP-pairs. In *Proceedings 26th International Symposium on Mathematical Foundations of Computer Science*, volume 2136 of *Lecture Notes in Computer Science*, pages 621–632. Springer-Verlag, Berlin, 2001.

[Raz94] A. Razborov. On provably disjoint NP-pairs. Technical Report TR94-006, Electronic Colloquium on Computational Complexity, 1994.

[Sad02] Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets of TAUT. *Theoretical Computer Science*, 288(1):181–193, 2002.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial-time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel82] A. Selman. Reductions on NP and p-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

[Sel94] A. Selman. A taxonomy on complexity classes of functions. *Journal of Computer and System Sciences*, 48:357–381, 1994.