Oracle with $P = NP \cap coNP$, but No Many-One Completeness in $UP$, $DisjNP$, and $DisjCoNP$

Anton Ehrmanntraut
Julius-Maximilians-Universität Würzburg, Germany

Fabian Egidy
Julius-Maximilians-Universität Würzburg, Germany

Christian Glaßer
Julius-Maximilians-Universität Würzburg, Germany

Abstract

We construct an oracle relative to which $P = NP \cap coNP$, but there are no many-one complete sets in $UP$, no many-one complete disjoint $NP$-pairs, and no many-one complete disjoint $coNP$-pairs.

This contributes to a research program initiated by Pudlák [33], which studies incompleteness in the finite domain and which mentions the construction of such oracles as open problem. The oracle shows that $NP \cap coNP$ is indispensable in the list of hypotheses studied by Pudlák. Hence one should consider stronger hypotheses, in order to find a universal one.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Oracles and decision trees; Theory of computation → Proof complexity

Keywords and phrases computational complexity, promise classes, proof complexity, complete sets, oracle construction

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.45


1 Introduction

Questions of the existence of complete sets in promise classes have a long history. They turned out to be difficult and remained open. Consider the following examples, where the questions are expressed as hypotheses.

$NP \cap coNP$ : $NP \cap coNP$ does not contain many-one complete sets [23]
$UP$ : $UP$ does not contain many-one complete sets [22]
$CON$ : $p$-optimal proof systems for $TAUT$ do not exist [26]
$SAT$ : $p$-optimal proof systems for $SAT$ do not exist [15]
$TFNP$ : $TFNP$ does not contain many-one complete problems [27]
$DisjNP$ : $DisjNP$ does not contain many-one complete pairs [34]
$DisjCoNP$ : $DisjCoNP$ does not contain many-one complete pairs [28, 32]

So far, the following implications are known: $DisjNP \Rightarrow CON$ [34], $UP \Rightarrow CON$ [25], $DisjCoNP \Rightarrow TFNP$ [33], $TFNP \Rightarrow SAT$ [5, 33], and $NP \cap coNP \Rightarrow CON \lor SAT$ [25]. This raises the question of whether further implications are provable with the currently available means. Thanks to a work by Pudlák [33], this question recently gained momentum. In fact, Pudlák’s interest goes beyond: He initiated a research program to find a general principle from which the remaining hypotheses follow as special cases. This is motivated by the study of incompleteness in the finite domain, since these hypotheses can either be expressed as the non-existence of complete elements in promise classes or as statements about the unprovability of sentences of some specific form in weak theories.
Pudlák [33] states as open problem to construct oracles that show that the relativized conjectures are different or show that they are equivalent. Such oracles have been constructed by Verbitskii [38], Glaßer et al. [18], Khaniki [24], Dose [10, 11, 9], and Dose and Glaßer [12].

The restriction to relativizable proofs arises from the following idea: We consider the mentioned hypotheses as conjectures, hence we expect that they are equivalent. In this situation we are not primarily concerned with the question of whether two hypotheses are equivalent, but rather whether their equivalence can be recognized with the currently available means. An accepted formalization of this is the notion of relativizable proofs.

**Our Contribution.** We construct an oracle relative to which the following holds: UP, DisjNP, DisjCoNP, but P = NP \cap coNP, which implies \neg NP \cap coNP. Hence there is no relativizable proof for NP \cap coNP, even if we simultaneously assume all remaining hypotheses we mentioned so far. This demonstrates that NP \cap coNP is indispensable in the list of currently viewed hypotheses and suggests to broaden the focus and include stronger statements.

Pudlák [33] ranks NP \cap coNP as a plausible conjecture that is apparently incomparable with CON and TFNP. Our oracle supports this estimation, as it rules out relativizable proofs for “CON \Rightarrow NP \cap coNP” and “TFNP \Rightarrow NP \cap coNP.” By Dose [10, 11], the same holds for the converse implications. Overall, we recognize a strong independence between NP \cap coNP and all remaining hypotheses:

(i) There does not exist a relativizable proof for NP \cap coNP, even if we simultaneously assume all remaining hypotheses.
(ii) There exists a relativizable proof for the implication NP \cap coNP \Rightarrow CON \lor SAT [25].

But there does not exist a relativizable proof showing that NP \cap coNP implies one of the remaining hypotheses [10, 11].

Our oracle combines several separations with the collapse P = NP \cap coNP. This leads to conclusions on the independence of the statement P \neq NP \cap coNP from typical assumptions. For instance, the oracle shows that P \neq NP \cap coNP cannot be proved by relativizing means, even under the strong but likely assumption UP \land DisjNP \land DisjCoNP.

Further characteristics of our oracle are, for example, NE \neq coNE, NPMV \subseteq c NPSV, and the shrinking and separation properties do not hold for NP and coNP. Corollary 10 presents a list of additional properties.

**Open Questions.** Currently, for almost every pair A, B of hypotheses, we either know a relativizable proof for the implication A \Rightarrow B, or we know an oracle relative to which A \land \neg B. Only three cases are left: (1) UP \Rightarrow DisjNP, (2) TFNP \Rightarrow DisjCoNP, and (3) SAT \Rightarrow TFNP. This leads to the following task for future research: Prove these implications or construct oracles relative to which they do not hold.

**Background on Connections Between Promise Classes and Proof Systems.** Informally, promise classes are complexity classes that are characterized by machines that satisfy certain properties. Usually, these properties are hard or even impossible to validate. Thus, when working with an element of a promise class, one has to trust the promise that the respective machine has said property. We are mainly interested in the following well-studied promise classes: The class of disjoint NP-pairs DisjNP = \{(A, B) | A, B \in NP, A \cap B = \emptyset\} [35, 21], the class of disjoint coNP-pairs DisjCoNP [14, 15] (defined respectively), the class of sets accepted by nondeterministic polynomial-time machines with at most one accepting computation path UP [37], the class NP \cap coNP [13], and the class of all total polynomial search problems TFNP [27]. As an example, the machines characterizing UP promise that on every input,
they either reject on all computation paths, or accept on exactly one computation path. Furthermore, we are interested in proof systems defined by Cook and Reckhow [8], especially optimal and p-optimal proof systems for the set of satisfiable formulas SAT, for the set of tautologies TAUT.

The connections between propositional proof systems and promise classes have been studied intensively. Krajíček and Pudlák [26] linked propositional proof systems (and thus the hypothesis CON) to standard complexity classes by proving that NE = coNE implies the existence of optimal propositional proof systems and E = NE implies the existence of p-optimal propositional proof systems. These results were subsequently improved by Köbler, Messner, and Torán [25].

Glaßer, Selman, and Sengupta [17] give several characterizations of DisjNP. Some characterizations use different notions of reducibility while others use the existence of \( \leq_{\text{m}} \)-complete functions in NPSV and the uniform enumerability of disjoint NP-pairs. Glaßer, Selman, and Zhang [19, 20] connect propositional proof systems to disjoint NP-pairs. They prove that the degree structure of DisjNP and of all canonical disjoint pairs of propositional proof systems is the same. Beyersdorff [1, 2, 3, 4] and Beyersdorff and Sadowksi [6] investigate further connections between disjoint NP-pairs and propositional proof systems.

Pudlák [30, 31, 33] draws connections between the finite consistency problem, proof systems, and promise classes like DisjNP and TFNP. Moreover, he asks for oracles that separate hypotheses regarding proof systems and promise classes. Several oracles have been constructed since Pudlák formulated his research questions. Concerning the listed hypotheses, Figure 1 summarizes all known (relativizing) implications and implications that do not hold relative to some oracle.
The paper is organized as follows: Section 2 defines the complexity classes mentioned above and presents our notations. Section 3 contains the oracle construction: the first part defines the construction, the second part proves that it is well-defined, and the last part shows the claimed properties.

2 Preliminaries

Basic Notation. Throughout this paper, let Σ be the alphabet {0, 1}. The set Σ* denotes the set of finite words over Σ. The set Σω denotes the set of ω-infinite words, i.e., the ω-infinite sequences of characters from Σ. Let Σ≤n := {w ∈ Σ* | |w| ≤ n}. For a word w ∈ Σ*∪Σω, we denote with w(i) the i-th character of w for 0 ≤ i < |w| ≤ ω. We write v ⊆ w when v is a prefix of w, that is, |v| ≤ |w| and v(i) = w(i) for all 0 ≤ i < |v|. Accordingly, v ⊊ w when v ⊆ w and v ̸= w. The empty word is denoted by ε. For a finite set A ⊆ Σ*, we define f(A) := |Σ< |w| | w ∈ A|.

Let N denote the set of non-negative integers, and N+ the set of positive integers. We say that two sets X and Y agree on set Z when X ∩ Z = Y ∩ Z.

The finite words Σ* can be linearly ordered by their quasi-lexicographic (i.e., “shortlex”) order ≺lex, uniquely defined by requiring 0 ≺lex 1. Under this definition, there is a unique order-isomorphism between (Σ*, ≺lex) and (N, <), which induces a polynomial-time computable, polynomial-time invertible bijection between Σ* and N. Hence, we can transfer the notations, relations, and operations for Σ* to N and vice versa. In particular, |n| denotes the length of the word represented by n ∈ N. By definition of ≺lex, whenever a ≤ b, then |a| ≤ |b|. We eliminate the ambiguity of the expressions 0l and 1l by always interpreting them over Σl. Moreover, < denotes both the less-than relation for natural numbers and the quasi-lexicographic order ≺lex for finite words. Similarly for ≤ and ≾lex. From the properties of order-isomorphism, this is compatible with the above identification of words and numbers.

Complexity Classes. We understand P (resp., NP) as the usual complexity class of languages decidable by a deterministic (resp., nondeterministic) polynomial-time Turing machine. The class FP refers to the class of total functions that can be computed by a deterministic polynomial-time Turing transducer [29]. Valiant [37] defined UP as the set of all languages that can be recognized by a nondeterministic polynomial-time machine that, on every input, accepts on at most one computation path. For a complexity class C we define coC := {A | A ∈ C} as the complementary complexity class of C. Between sets of words, we employ the usual polynomial-time many-one reducibility: A ≤m B if there exists an f ∈ FP such that x ∈ A ⇔ f(x) ∈ B. The usual notion of ≤m-completeness and -hardness follows.

A disjoint NP-pair is a pair (A, B) of disjoint sets in NP. Selman [35] and Grollmann and Selman [21] defined the class DisjNP as the set of disjoint NP-pairs. The classes DisjCoNP [14, 15], DisjUP, and DisjCoUP are defined similarly. Between two pairs, we employ the following related notion of reducibility [34, 18]: Let (A, B) and (C, D) be two disjoint pairs. We say that (A, B) is polynomial-time many-one reducible to (C, D), denoted by (A, B) ≤mp (C, D), if there is a function h ∈ FP such that h(A) ⊆ C and h(B) ⊆ D. The terms ≤mp-completeness and -hardness also follow directly from this definition of reduction.

Proof Systems. We use the notion of proof systems for sets by Cook and Reckhow [8]: A function f ∈ FP is called a proof system for img(f). Specifically, a proof system f for TAUT is a propositional proof system. We say that a proof system g is (p-)simulated by a proof system f, denoted by f ≤g (resp., f ≤p g), if there exists a total function π (resp.,
\( \pi \in \text{FP} \) and a polynomial \( p \) such that \( |\pi(x)| \leq p(|x|) \) and \( f(\pi(x)) = g(x) \) for all \( x \). We call a proof system \( f \) \((p-)optimal\) for the set \( \text{img}(f) \), if \( g \leq f \) (resp., \( g \leq^p f \)) for all \( g \in \text{FP} \) with \( \text{img}(g) = \text{img}(f) \).

**Relativizations.** We can relativize each complexity and function class to some oracle \( O \), by equipping all machines corresponding to the respective class with oracle access to \( O \). That is, e.g., \( \text{UP}^O := \{ L(M^O) \mid M \text{ is a nondeterministic polynomial-time oracle Turing machine, and for all inputs } x, M^O(x) \text{ accepts on at most one path } \} \). The classes \( \text{P}^O \), \( \text{NP}^O \) and so on are defined similarly. We can also relativize our notions of reducibility by using functions from \( \text{FP}^O \) instead of \( \text{FP} \). In other words, we allow the reduction functions to access the oracle in relativized instances. This results in polynomial-time many-one reducibilities relative to an oracle \( O \), which we denote as \( \leq^m_{\text{in}}^O \) for sets and \( \leq^m_{\text{pp}}^O \) for pairs of disjoint sets. In the same way, we can relativize \((p-)simulation\) of proof systems to some oracle \( O \), and denote the relativized simulation as \( \leq^s_O \) resp. \( \leq^p_O \). When it is clear from context that some statements refer to the relativized ones relative to some fixed oracle \( O \), we sometimes omit the indication of \( O \) in the superscripts.

We define \( p_i(n) := n^i + i \). Let \( \{M_i\}_{i \in \mathbb{N}} \) and \( \{F_i\}_{i \in \mathbb{N}} \) be, respectively, standard enumerations of nondeterministic polynomial-time (oracle) Turing machines resp. deterministic polynomial-time (oracle) Turing transducers, having the property that runtime of \( M_i,F_i \) is bounded by \( p_i \) relative to any oracle. Note that \( \{L(M_i^O) \mid i \in \mathbb{N}\} = \text{NP}^O \), \( \{F_i^O \mid i \in \mathbb{N}\} = \text{FP}^O \).

**Specific Notation Used in our Oracle Construction.** We now take on the notations proposed by Dose and Glaßer [12] designed for the construction of oracles. The domain of definition, image, and support for partial function \( t : A \rightarrow N \) are defined as \( \text{dom}(t) := \{x \in A \mid t(x) \text{ defined}\} \), \( \text{img}(t) := \{t(x) \mid x \in A, t(x) \text{ defined}\} \), \( \text{supp}(t) := \{x \in A \mid t(x) \text{ defined and } t(x) > 0\} \). We say that \( t \) is injective on its support if, for any \( a,b \in \text{supp}(t) \), \( t(a) = t(b) \) implies \( a = b \). If \( t \) is not defined at point \( x \), then \( t \cup \{x \mapsto y\} \) denotes the extension \( t' \) of \( t \) at \( x \) has value \( y \) and satisfies \( \text{dom}(t') = \text{dom}(t) \cup \{x\} \).

For a set \( A \), we denote with \( A(x) \) the characteristic function at point \( x \), i.e., \( A(x) \) is 1 if \( x \in A \), and 0 otherwise. We can identify an oracle \( A \subseteq \mathbb{N} \) with its characteristic \( \omega \)-word \( A(0)A(1)A(2) \cdots \) over \( \Sigma^\omega \). In this way, \( A(i) \) denotes both the characteristic function at point \( i \) and the \( i \)-th character of its characteristic word. Similarly, for a finite word \( w \in \Sigma^* \), we also understand \( w \) as the set \( \{i \mid w(i) = 1\} \) and, e.g., we write \( A = w \cup B \) where \( A \) and \( B \) are sets. (However, we understand \( |w| \) as the length of the word \( w \), and not the cardinality of set \( \{i \mid w(i) = 1\} \).) Thus, a finite word \( w \) describes an oracle which is partially defined, i.e., only defined for natural numbers (or equivalently words) \( x < |w| \). Being able to interpret a word \( w \) as a set and partial oracle is very useful for the oracle construction. In most construction steps we decide the membership of the smallest undefined word of a partial oracle \( w \), which is simply \( |w| \). This gives access to very concise notation.

In particular, for oracle machines \( M \), the notation \( M^w(x) \) refers to \( M^{(1)\upharpoonright w(i) = 1}(x) \) (that is, oracle queries that \( w \) is not defined for are negatively answered). This also allows us to define the following notion: we say that \( M^w(x) \) is \textit{define} if all queries on all computation paths are \( < |w| \) (or equivalently: \( w(q) \) is defined for all queries \( q \) on all computation paths); we say that \( M^w(x) \) \textit{definitely accepts} (resp., \textit{definitely rejects}) if \( M^w(x) \) is definite and accepts (resp., rejects). Intuitively, the term definite describes computations that do not change when extending the respective oracle, because the queries are too short. This allows the following observation:
45:6 Oracle with $P = \text{NP} \cap \text{coNP}$ but No Completeness in UP, DisjNP, DisjCoNP

Observation 1.

(i) When $M^w(x)$ is a definite computation, and $v \supseteq w$, then $M^v(x)$ is definite. Computation $M^v(x)$ accepts if and only if $M^w(x)$ accepts.

(ii) When $w$ is defined for all words of length $p(x)$, then $M^w(x)$ is definite.

(iii) When $M^w(x)$ accepts on some computation path with set of oracle queries $Q$, and $w, v$ agree on $Q$, then $M^v(x)$ accepts on the same computation path and with the same set of oracle queries $Q$.

For an oracle $w$, a transducer $F$, and a machine $M$, we occasionally understand the notation $M^w(F^w(x))$ as the single computation of the machine $M \circ F$ on input $x$ relative to $w$. Consequently, we say that $M^u(F^w(x))$ definitely accepts (resp., rejects) when $M \circ F$ definitely accepts (resp., rejects) input $x$ relative to $w$.

In our oracle construction, we want to injectively reserve and assign countably infinitely many levels, that are, words of same length $n$, for a countably infinite family of witness languages, with increasingly large gaps. For this, let $e(0) := 2, e(i) := 2^{e(i-1)}$. There is a polynomial-time computable, polynomial-time invertible injective function $f$, mapping $(m, h) \in \mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. Now define $H_m := \{e(f(m, h)) | h \in \mathbb{N}\}$ as the set of levels reserved for witness language $m$. This definition ensures

Observation 2.

(i) The set $H_m$ is countably infinite, a subset of the even numbers, and all $H_0, H_1, \ldots$ are pairwise disjoint.

(ii) The sequence $\min H_0, \min H_1, \ldots$ is unbounded.

(iii) When $n \in H_m$, then $n < n' < 2^n$ implies $n' \notin H_0, H_1, \ldots$.

(iv) Every set $H_m \subseteq \mathbb{P}$ for all $m \in \mathbb{N}$.

3 Oracle Construction

We are primarily interested in an oracle $O$ with the property that relative to that oracle, UP, DisjNP, DisjCoNP, and $\neg \text{NP} \cap \text{coNP}$ hold, but our construction yields the following slightly stronger statements:

(i) NP $\cap$ coNP $\subseteq \mathbb{P}$ (implying $\neg \text{NP} \cap \text{coNP}$).

(ii) DisjNP does not contain $\leq_{\text{pp}}$-hard pairs for DisjUP (implying DisjNP).

(iii) UP does not contain $\leq_{\text{pp}}$-complete languages (i.e., UP).

(iv) DisjCoNP does not contain $\leq_{\text{pp}}$-hard pairs for DisjCoUP (implying DisjCoNP).

Given a (possibly partial) oracle $O$ and $m \in \mathbb{N}$, we define the following witness languages:

\[
\begin{align*}
A_m^O & := \{0^n | n \in H_m, \text{there exists } x \in \Sigma^n \text{ such that } x \in O \text{ and } x \text{ ends with } 0\} \\
B_m^O & := \{0^n | n \in H_m, \text{there exists } x \in \Sigma^n \text{ such that } x \in O \text{ and } x \text{ ends with } 1\} \\
C_m^O & := \{0^n | n \in H_m, \text{there exists } x \in \Sigma^n \text{ such that } x \in O\} \\
D_m^O & := \{0^n | n \in H_m, \text{for all } x \in \Sigma^n, x \in O \Rightarrow x \text{ ends with } 0\} \\
E_m^O & := \{0^n | n \in H_m, \text{for all } x \in \Sigma^n, x \in O \Rightarrow x \text{ ends with } 1\}
\end{align*}
\]

Their purpose is to be a “witness” that an element of DisjNP (resp., UP, DisjCoNP) is not complete by admitting no reduction to this element. This only works if the witness languages themselves belong to the respective classes. The following observation shows how the membership of the witness languages to the respective classes depends on the oracle.
\textbf{Observation 3.}\n
(i) If for all \(n \in H_m\), \(|O \cap \Sigma^n| \leq 1\), then \((A^O_m, B^O_m)\) is in \text{DisjUP}^O\), and \(C^O_m\) is in \(\text{UP}^O\).

(ii) If for all \(n \in H_m\), \(O \cap \Sigma^n\) contains at least one word but not two words with the same parity, (i.e., there exists \(\alpha \in \Sigma^{n-1}0\), \(\beta \in \Sigma^{n-1}1\) such that the set \(O \cap \Sigma^n\) is equal to \{\(\alpha\) or \{\(\beta\) or \{\(\alpha, \beta\)\}, then \((D^O_m, E^O_m)\) is in \text{DisjCoUP}^O\).

**Preview of the Construction.** The construction is quite technical, since the oracle has to satisfy several properties which simultaneously demand structure (i.e., property (i)) and freedom (i.e., properties (ii), (iii) and (iv)) during the construction. This leads to several dependencies and special cases that need to be addressed, mostly by combinatorial arguments and various extensions of the oracle constructed so far. To keep track of the progress of the construction, it is divided into tasks corresponding to the desired properties (i)–(iv). Each task contributes to the goal of satisfying its corresponding property.

1. **Work towards** \(P = \text{NP} \cap \text{coNP}\): For all \(a \neq b\), the construction tries to achieve that \(M_a, M_b\) do not accept complementarily. (Accepting complementary should mean that for each input \(x\), precisely one of \(M_a(x)\) accepts and the other rejects.) If this is not possible, \((M_a, M_b)\) inherently accept complementarily, and thus \(L(M_a) \in \text{NP} \cap \text{coNP}\). Then, we start to encode into the oracle, whether \(M_a\) accepts some inputs or not. Thus, the final oracle will contain the encodings for almost all inputs, thus allowing to recover the accepting behavior of \(M_a\) and hence to decide \(L(M)\) in \(P\) using oracle queries.

2. **Work towards (ii)**, which implies \(\text{DisjNP}\): For all \(i \neq j\), the construction tries to achieve that \(M_i, M_j\) both accept some input \(x\), hence \(x \in L(M_i) \cap L(M_j)\) and \((L(M_i), L(M_j)) \notin \text{DisjNP}\). If this is not possible, \((M_i, M_j)\) inherently is a disjoint NP-pair. In this case, we fix some \(m\), make sure that \((A_m, B_m)\) is a disjoint UP-pair and diagonalize against every transducer \(F_r\), so that \(F_r\) does not realize the reduction \((A_m, B_m) \leq_{\text{PP}} (L(M_i), L(M_j))\). This is achieved by, (i) for all \(n \in H_m\), insert at most one word of length \(n\) into \(O\) (and thus \((A_m, B_m) \in \text{DisjUP}\)), and (ii) for every \(r\) there is an \(n \in H_m\) such that \(0^n \in A_m \) but \(M_j(F_r(0^n))\) rejects (or analogously \(0^n \in B_m \) but \(M_j(F_r(0^n))\) rejects).

3. **Work towards (iii)**, i.e., \(UP\): Try to make \(M_i\) accept on two separate paths. If this is not possible, then \(L(M_i)\) inherently is a UP-language. In this case, we fix some \(m\), make sure that \(C_m\) is a language in \(UP\) and diagonalize against every transducer \(F_r\) so that \(F_r\) does not realize the reduction \(C_m \leq_{\text{PP}} L(M_i)\). This is achieved by, (i) for all \(n \in H_m\), insert at most one word of length \(n\) into \(O\) (and thus \(C_m \in UP\)), and (ii) for every \(r\) there is an \(n \in H_m\) such that \(0^n \in C_m\) if and only if \(M_j(F_r(0^n))\) rejects.

4. **Work towards (iv)**, which implies \(\text{DisjCoNP}\): Try to achieve that \(M_i, M_j\) both reject some input \(x\), hence \(x \notin L(M_i) \cap L(M_j)\) and \((L(M_i), L(M_j)) \notin \text{DisjNP}\). If this is not possible, \((M_i, M_j)\) inherently is a disjoint coNP-pair. In this case, we fix some \(m\), make sure that \((D_m, E_m)\) is a disjoint coUP-pair and diagonalize against every transducer \(F_r\), so that \(F_r\) does not realize the reduction \((D_m, E_m) \leq_{\text{PP}} (L(M_i), L(M_j))\). This is achieved by, (i) for all \(n \in H_m\), insert at least one word of length \(n\) into \(O\) but not two words with same parity (and thus \((D_m, E_m) \in \text{DisjCoUP}\)), and (ii) for every \(r\) there is an \(n \in H_m\) such that \(0^n \in D_m\) but \(M_i(F_r(0^n))\) accepts (or analogously \(0^n \in E_m\) but \(M_j(F_r(0^n))\) accepts).

To these requirements, we assign the following symbols representing tasks: \(\tau^1_{a,b}, \tau^2_{i,j}, \tau^2_{i,j,r}, \tau^3_{i,j}, \tau^3_{i,j,r}\) for all \(a, b, i, j, r \in \mathbb{N}, i \neq j, a \neq b\). The symbol \(\tau^1_{a,b}\) represents the encoding or the destruction of \(\text{NP} \cap \text{coNP}\)-pairs. The symbol \(\tau^2_{i,j}\) represents the destruction of a disjoint \(\text{NP}\)-pair, \(\tau^3_{i,j}\), the diagonalization of that pair against transducer \(F_r\). Analogously for \(\text{UP}\) and \(\tau^4_{i,j}\), \(\tau^4_{i,j,r}\). Analogously for \(\text{DisjCoNP}\) and \(\tau^4_{i,j}\), \(\tau^4_{i,j,r}\).
For the coding, we injectively define the code word \(c(a, b, x) := 0^n1^{10^l}10^{p_1(x)}1\) with \(p = p_4(x) + p_3(x), l \in \mathbb{N}\) minimal such that \(l \geq 7/|c(a, b, x)|\) and \(c(a, b, x)\) has odd length. By this, a code word contains the word \(x\) as information and is padded to sufficient length. We call any word of the form \(c(\cdot, \cdot, \cdot)\) a code word. This ensures the following properties:

> **Claim 4.** For all \(a, b \in \mathbb{N}, x \in \Sigma^*\), the following holds:
> (i) \(|c(a, b, x)| \notin H_m\) for any \(m\).
> (ii) For fixed \(a, b\), the function \(x \mapsto c(a, b, x)\) is polynomial-time computable, and polynomial-time invertible with respect to \(|x|\).
> (iii) Relative to any oracle, the running times of \(M_a(x)\) and \(M_b(x)\) are both bounded by \(< |c(a, b, x)|/8\).
> (iv) For every partial oracle \(w \in \Sigma^*\), if \(c(a, b, x) \leq |w|\), then \(M_a^w(x)\) and \(M_b^w(x)\) are definite.

During the construction we successively add requirements that we maintain. To keep track of these requirements, we use a partial function \(t\). Define \(T\) as the set of all partial functions \(t\) mapping \(\tau^1_{a,b}, \tau^2_{i,j}, \tau^3_i, \tau^4_{i,j}, i \neq j, a \neq b\) to \(\mathbb{N}\). In fact, these requirements determine to a large extent how tasks are treated and are mainly responsible that the oracle satisfies the desired properties. To add a requirement in the construction, we can extend the function \(t\).

Define \(T\) as the set of all partial functions \(t\) mapping \(\tau^1_{a,b}, \tau^2_{i,j}, \tau^3_i, \tau^4_{i,j}, i \neq j, a \neq b\) to \(\mathbb{N}\), and \(\text{dom}(t)\) is finite, and \(t\) is injective on its support.

To now link the maintenance of the requirements with the oracle construction, we introduce the notion of validity. A partial oracle \(w \in \Sigma^*\) is called \(t\)-valid for \(t \in T\) if it satisfies the following properties:

- **V1** If \(\tau^1_{a,b} = 0\), then there exists an \(x\) such that \(M^w_a(x), M^w_b(x)\) both definitely accept or both definitely reject.

  (Meaning: if \(\tau^1_{a,b} = 0\), then for every extension of the oracle, \(M_a, M_b\) do not accept complementary.)

- **V2** If \(0 < t(\tau^1_{a,b}) \leq c(a, b, x) < |w|\), then \(M^w_a(x)\) is definite. Additionally, the computation \(M^w_a(x)\) accepts when \(c(a, b, x) \in w\), and rejects when \(c(a, b, x) \notin w\). When the above conditions are met by \(c(a, b, x)\) we sometimes refer to these code words as mandatory code words with respect to some \(t\)-valid partial oracle \(w\). Note that when the previous conditions are not met \((\tau^1_{a,b} \notin \text{dom}(t)\) or \(t(\tau^1_{a,b}) = 0\) or \(t(\tau^1_{a,b}) > c(a, b, x)\)) then the code word \((a, b, x)\) may be a member of oracle \(w\), independent of \(M_a, M_b\).

  (Meaning: if \(t(\tau^1_{a,b}) > 0\), then from \(t(\tau^1_{a,b})\) on, we encode \(L(M_a)\) into the oracle. That is, \(L(M^w_a) = \{(x \mid c(a, b, x) \in O) \cup \text{some finite set} \} \in P^O\).)

- **V3** If \(\tau^1_{a,b} = 0\), then there exists an \(x\) such that \(M^w_a(x), M^w_b(x)\) both definitely accept.

  (Meaning: if \(t(\tau^1_{a,b}) = 0\), then for every extension of the oracle, \((L(M_a), L(M_b)) \notin \text{DisjNP}\).)

- **V4** If \(t(\tau^2_{i,j}) = m > 0\), then for every \(n \in H_m\) it holds that \(|\Sigma^n \cap w| \leq 1\).

  (Meaning: if \(t(\tau^2_{i,j}) = m > 0\), then ensure that \((A_m, B_m) \in \text{DisjUP relative to the final oracle}\).)

- **V5** If \(t(\tau^2_{i,j}) = 0\), then there exists an \(x\) such that \(M^w_a(x)\) is definite and accepts on two different paths.

  (Meaning: if \(t(\tau^2_{i,j}) = 0\), then for every extension of the oracle, \(L(M_i) \notin \text{UP}\).)

- **V6** If \(t(\tau^3) = m > 0\), then for every \(n \in H_m\) it holds that \(|\Sigma^n \cap w| \leq 1\).

  (Meaning: if \(t(\tau^3) = m > 0\), then ensure that \(C_m \in \text{UP relative to the final oracle}\).)

- **V7** If \(\tau^4_{i,j} = 0\), then there exists an \(x\) such that \(M^w_a(x), M^w_b(x)\) both definitely reject.

  (Meaning: if \(t(\tau^4_{i,j}) = 0\), then for every extension of the oracle, \((L(M_i), L(M_j)) \notin \text{DisjCoNP}\).)
If $t(\tau_{a,b}^1) = m > 0$, then for every $n \in H_m$ it holds that all words in $\Sigma^n \cap w$ have pairwise different parity. If additionally $w$ is defined for all words of length $n$, then $|\Sigma^n \cap w| > 0$. (Meaning: if $t(\tau_{a,b}^1) = m > 0$, then ensure that $(D_m, E_m) \in \text{DisjCoUP}$ relative to the final oracle.)

Intuitively, a $t$-valid oracle is a possibly partial oracle which has the desired properties (i)–(iv) “partially satisfied”. This notion of validness helps in the oracle construction, since our oracle is defined inductively, the induction step deals with a partial oracle and therefore $t$-validity fits great as part of an induction hypothesis which states that the partial oracle is constructed properly so far. Observe that $V_2, V_4, V_6, V_8$ do not (pairwise) contradict each other, since $t$ is injective on its support and all $H_1, H_2, \ldots$ are pairwise disjoint, by Observation 2(i).

Also observe that $V_2$ and $V_4$ (resp., $V_2$ and $V_6$, $V_2$ and $V_8$) do not contradict each other, as $c(\cdot, \cdot, \cdot)$ has odd length, but all $n$ in all $H_m$ are even by Observation 2(ii).

Oracle Construction. Let $T$ be a countable enumeration of

$$
\{\tau_{a,b}^1 | a, b \in \mathbb{N}, a \neq b\} \cup \{\tau_{i,j}^2 | i, j \in \mathbb{N}, i \neq j\} \cup \{\tau_{i,j,r}^3 | i, j, r \in \mathbb{N}, i \neq j\}
\cup \{\tau_{i,r}^3 | i, r \in \mathbb{N}\}
\cup \{\tau_{i,j,r}^4 | i, j, r \in \mathbb{N}, i \neq j\}
$$

with the property that $\tau_{i,j}^2$ appears earlier than $\tau_{i,j,r}^2$, $\tau_{i,r}^3$ appears earlier than $\tau_{i,s}^3$, $\tau_{i,j}^4$ earlier than $\tau_{i,j,r}^4$.

We inductively define an infinite sequence $\{(w_s, t_s)\}_{s \in \mathbb{N}}$, where the $s$-th term of the sequence is a pair $(w_s, t_s)$ of a partial oracle and a function in $T$. We call the $s$-th term the stage $s$. We ensure that for all $s \in \mathbb{N}$, $w_s$ is a $t_s$-valid partial oracle.

In each stage, we treat the smallest task in the order specified by $T$, and after treating a task we remove it and possibly other higher tasks from $T$. In the next stage, we continue with the next task not already removed from $T$. (In every stage, there always exists a task not already removed, as we never remove all remaining tasks from $T$ in any stage.)

We start with the nowhere defined function $t_0 \in T$ and the $t_0$-valid oracle $w_0 := \varepsilon$ as 0-th stage. Then we begin treating the tasks.

Thus, for stage $s > 0$, we have that $w_0, w_1, \ldots, w_{s-1}$ and $t_0, t_1, \ldots, t_{s-1}$ are defined. With this, we define the $s$-th stage $(w_s, t_s)$ such that (a) $w_{s-1} \subseteq w_s$, and $t_s \in T$ is a (not necessarily strict) extension of $t_{s-1}$, and (b) $w_s$ is $t_s$-valid, and (c) the earliest task $\tau$ still in $T$ is treated and removed in some way.

So for each task we strictly extend the oracle and are allowed to add more requirements, by extending the valid function, that have to be maintained in the further construction. Finally, we choose $O := \bigcup_{s \in \mathbb{N}} w_s$. (Note that $O$ is totally defined since in each step we strictly extend the oracle.) Also, every task in $T$ is assigned some stage $s$ where it was treated (or removed from $T$).

We now define stage $s > 0$, which starts with some $t_{s-1} \in T$ and a $t_{s-1}$-valid oracle $w_{s-1}$ and treats the first task that still is in $T$ choosing an extension $t_s \in T$ of $t_{s-1}$ and a $t_s$-valid $w_s \supseteq w_{s-1}$. Let us recall that each task is immediately deleted from $T$ after it is treated. There are seven cases depending on the form of the task that is treated in stage $s$:

- Task $\tau_{a,b}^1$. Let $t' := t_{s-1} \cup \{\tau_{a,b}^1 \mapsto 0\}$. If there exists a $t'$-valid $v \supseteq w_{s-1}$, then assign $t_s := t'$ and let $w_s := v$.

Otherwise, let $t_s := t_{s-1} \cup \{\tau_{a,b}^1 \mapsto n\}$ with $n \in \mathbb{N}^+$ sufficiently large such that $n > |w_s|$, $\max_g(t_{s-1})$. Thus $t_s$ is injective on its support, and $w_{s-1}$ is $t_s$-valid. Let $w_s := w_{s-1}y$ with $y \in \{0, 1\}$ such that $w_s$ is $t_s$-valid. Lemma 5 shows that such $y$ does indeed exist.

MFCS 2022
Task $\tau_{i,j}^2$: Let $t' := t_{s-1} \cup \{\tau_{i,j}^2 \mapsto 0\}$. If there exists $t'$-valid $v \supseteq w_{s-1}$, then assign $t_s := t'$ and $w_s := v$. Besides task $\tau_{i,j}^2$, also remove all tasks $\tau_{i,j,0}, \tau_{i,j,1}, \ldots$ from $T$.

Otherwise, let $t_s := t_{s-1} \cup \{\tau_{i,j}^3 \mapsto m\}$ with $m \in \mathbb{N}^+$ sufficiently large such that $m \not\in \text{img}(t_{s-1})$ and that $w_{s-1}$ defines no word of length $\min H_m$. Thus $t_s$ is injective on its support, and $w_{s-1}$ is $t_s$-valid. Let $w_s := w_{s-1}y$ with $y \in \{0, 1\}$ such that $w_s$ is $t_s$-valid. Again, Lemma 5 shows that such $y$ does indeed exist.

(Meaning: try to ensure that $M_i, M_j$ do not accept disjointly, cf. V5. If that is impossible, choose a sufficiently large “fresh” $m$ and require for the further construction that $(A_m, B_m) \in \text{DisjUP}$ (cf. V4). The treatment of the tasks $\tau_{i,j,0}^2, \tau_{i,j,1}^2, \ldots$ defined below makes sure that $(A_m, B_m)$ cannot be reduced to $(L(M_i), L(M_j))$.)

Task $\tau_{i,j}^3$: Defined symmetrically to task $\tau_{i,j}^2$. (Meaning: try to ensure that $M_i, M_j$ do not accept disjointly, cf. V7. If that is impossible, choose a sufficiently large “fresh” $m$ and require for the further construction that $(D_m, E_m) \in \text{DisjCoUP}$ (cf. V8). The treatment of the tasks $\tau_{i,j,0}^3, \tau_{i,j,1}^3, \ldots$ defined below makes sure that $(D_m, E_m)$ cannot be reduced to $(L(M_i), L(M_j))$.)

Task $\tau_{i,j}^4$: We have $t_{s-1} (\tau_{i,j}^4) = m \in \mathbb{N}^+$. Let $t_s := t_{s-1}$ and choose a $t_s$-valid $w_s \supseteq w_{s-1}$ such that there is some $n \in \mathbb{N}$ and at least one of the following holds:

- $0^n \in A_m$ for all $v \supseteq w_s$ and $M_i^{w_s} (F_r^{w_s} (0^n))$ definitely rejects.
- $0^n \in B_m$ for all $v \supseteq w_s$ and $M_j^{w_s} (F_r^{w_s} (0^n))$ definitely rejects.

In Theorem 6 we show that such $w_s$ does exist.

(Meaning: ensure that $F_r$ does not reduce $(A_m, B_m)$ to $(L(M_i), L(M_j))$.)

Task $\tau_{i,j}^5$: We have $t_{s-1} (\tau_{i,j}^5) = m \in \mathbb{N}^+$. Let $t_s := t_{s-1}$ and choose a $t_s$-valid $w_s \supseteq w_{s-1}$ such that there is some $n \in \mathbb{N}$ and at least one of the following holds:

- $0^n \notin C_m$ for all $v \supseteq w_s$ and $M_i^{w_s} (F_r^{w_s} (0^n))$ definitely rejects.
- $0^n \notin C_m$ for all $v \supseteq w_s$ and $M_j^{w_s} (F_r^{w_s} (0^n))$ definitely accepts.

In Theorem 7 we show that such $w_s$ does exist.

(Meaning: ensure that $F_r$ does not reduce $C_m$ to $L(M_i)$.)

Task $\tau_{i,j}^6$: Defined symmetrically to $\tau_{i,j}^5$. Choose a $t_r$-valid $w_s \supseteq w_{s-1}$ such that for some $n \in \mathbb{N}$, one of the two holds:

- $0^n \in D_m$ for all $v \supseteq w_s$ and $M_i^{w_s} (F_r^{w_s} (0^n))$ definitely accepts.
- $0^n \in E_m$ for all $v \supseteq w_s$ and $M_j^{w_s} (F_r^{w_s} (0^n))$ definitely accepts.

In Theorem 8 we show that such $w_s$ does exist.

(Meaning: ensure that $F_r$ does not reduce $(D_m, E_m)$ to $(L(M_i), L(M_j))$.)
Observe that \( t_s \) is always defined to be in \( T \). Remember that the treated task is immediately deleted from \( T \). This completes the definition of stage \( s \), and thus, the entire sequence \( \{(w_s, t_s)\}_{s \in \mathbb{N}} \). We now show that this construction is indeed possible. The proofs of the theorems/lemma announced in the definition are either roughly sketched or omitted. For detailed proofs, we refer to the full version of the paper. It is not difficult to see that a valid oracle can be extended by one bit such that it remains valid:

**Lemma 5.** Let \( s \in \mathbb{N}, (w_0, t_0), \ldots, (w_s, t_s) \) defined, and let \( w \in \Sigma^* \) be a \( t_s \)-valid oracle with \( w \supseteq w_s \), and \( z := |w| \). (Think of \( z \) as the next word we need to decide its membership to the oracle, i.e., \( z \not\in w_0 \) or \( z \in w_1 \).) Then there exists \( y \in \{0, 1\} \) such that \( wy \) is \( t_s \)-valid. Specifically:

1. If \( z = c(a, b, x) \) and \( 0 < t_s(\tau_{a,b}^1) \leq z \), then \( w1 \) is \( t_s \)-valid if \( M_w^w(x) \) accepts (or when \( M_w^w(x) \) rejects), and \( w0 \) is \( t_s \)-valid if \( M_w^w(x) \) rejects (or when \( M_w^w(x) \) accepts).
2. If there exists \( \tau = \tau_{i,j}^2 \) with \( m = t_s(\tau) > 0 \) and \( n \in H_m \) such that \( |z| = n \), \( w \cap \Sigma^n \neq \emptyset \), then \( w0 \) is \( t_s \)-valid.
3. If there exists \( \tau_{i,j}^3 \), \( m = t_s(\tau_{i,j}^3) > 0 \) and \( n \in H_m \) such that \( |z| = n \) and there is some other word \( x \in w \cap \Sigma^n \) with same parity as \( z \), then \( w0 \) is \( t_s \)-valid. (Meaning: if we are on a level \( n \) belonging to a DisjUP-pair or a UP-language, ensure that there is no more than one word on that level.)
4. If there exists \( \tau_{i,j}^4 \), \( m = t_s(\tau_{i,j}^4) > 0 \) and \( n \in H_m \) such that \( |z| = n \) and there is some other word \( x \in w \cap \Sigma^n \) with same parity as \( z \), then \( w0 \) is \( t_s \)-valid. (Meaning: if we finalize level \( n \) belonging to a DisjCoUP-pair, ensure that on that level, there are no two words with the same parity.)
5. In all other cases, \( w0 \) and \( w1 \) are \( t_s \)-valid.

Lemma 5 shows that the construction is possible for the tasks \( \tau_{a,b}^1, \tau_{i,j}^2, \tau_{i,j}^3 \) and \( \tau_{i,j}^4 \). Now we show that the construction is possible for \( \tau_{i,j}^2, \tau_{i,j}^3 \) and \( \tau_{i,j}^4 \), respectively. We first consider task \( \tau_{i,j}^2 \).

**Theorem 6.** Let \( s \in \mathbb{N}, (w_0, t_0), \ldots, (w_{s-1}, t_{s-1}) \) defined. Consider task \( \tau_{i,j}^2 \).

Suppose that \( t_s = t_{s-1}, t_s(\tau_{i,j}^2) = m > 0 \). Then there exists a \( t_s \)-valid \( w \supseteq w_{s-1} \) and \( n \in \mathbb{N} \) such that the two holds:

1. \( 0^n \in A_w^w \) for all \( v \supseteq w \) and \( M_w^w(F_v^w(0^n)) \) definitely rejects.
2. \( 0^n \in B_w^w \) for all \( v \supseteq w \) and \( M_w^w(F_v^w(0^n)) \) definitely rejects.

**Proof sketch.** We prove the Theorem by contradiction. Let \( \hat{s} < s \) be the stage that treated \( \tau_{i,j}^2 \) with \( t_{\hat{s}} = t_{s-1} \cup \{\tau_{i,j}^2 \mapsto m\} \) and \( m > 0 \). We construct a suitable alternative oracle \( u' \supseteq w_{s-1} \), which is valid with respect to \( t' := t_{\hat{s}} \cup \{\tau_{i,j}^2 \mapsto 0\} \). Then, by definition, we obtain that \( u' \) is one possible \( t' \)-valid extension of \( w_{s-1} \) in stage \( \hat{s} \), hence \( t_{\hat{s}} = t' \) and \( t_s(\tau_{i,j}^2) = m > 0 \) in the hypothesis of this Theorem 6.

Assume that (i) and (ii) do not hold. Fix some sufficiently large \( n \in H_m \). This is some level that belongs to the witness NP-pair \( (A_m, B_m) \). For every \( \xi \in \Sigma^n \), one can provisionally keep extending \( w_{s-1} \) bitwise with Lemma 5 while inserting precisely the word \( \xi \) into level \( n \). Continue extending until a sufficiently long but fixed length \( n' \) such that \( M_\alpha(F_\alpha(0^n')) \) and \( M_\beta(F_\beta(0^n')) \) are definite (relative to any oracle), and call the resulting oracle \( \xi_{u_\alpha} \). By construction, this oracle is \( t_s \)-valid. By assumption, when \( \alpha \in \Sigma^{n-10} \), then \( M_\alpha(F_\alpha(0^n')) \) accepts relative to \( u_\alpha \), and when \( \beta \in \Sigma^{n-11} \), then \( M_\beta(F_\beta(0^n')) \) accepts relative to \( u_\beta \).
We want to maintain these accepting computations relative to an oracle containing, in level \( n \), exactly one \( \alpha \in \Sigma^n \) and one \( \beta \in \Sigma^n \). The accepting behavior for any such computation, say \( M_i(F_i(0^n)) \) relative to \( u_\alpha \), depends on the oracle queries posed on that path. However, this computation might also query certain mandatory code words \( c(a, b, x) \), whose memberships depend on further, shorter queries of computations \( M_\beta(x) \), \( M_\delta(x) \). Continuing this recursively, we obtain a set of queries \( Q^+_\alpha \) the original computation \( M_i(F_i(0^n)) \) transitively depends on.

Similarly, for each \( \beta \in \Sigma^n \) we can define a set of queries \( Q^+_\beta \) an accepting path of the computation \( M_j(F_j(0^n)) \) relative to \( u_\beta \) depends on. One can verify that \( Q^+_\alpha \) and \( Q^+_\beta \) only have polynomially many elements.

The crucial idea that completes the proof is to find suitable \( t_{s-1} \)-valid oracles \( u' \) whose memberships depend on further, shorter queries of computations \( t_{s-1} \)-valid oracles \( u \) and \( u_\alpha \) agree on \( Q^+_\alpha \) and \( u_\beta \) agrees on \( Q^+_\beta \). By assumption and Observation 1(iii), this means that both \( M_i(F_i(0^n)) \) and \( M_j(F_j(0^n)) \) definitely accept relative to \( u' \). Thus \( u' \) is also \( t \)-valid, as desired.

With only slight modifications, one can give the same Theorem concerning tasks \( r_{i,j} \). We omit the specific details.

\( \blacktriangle \)

**Theorem 7.** Let \( s \in \mathbb{N}^+ \), \( (w_0, t_0), \ldots, (w_{s-1}, t_{s-1}) \) defined. Consider task \( r_{i,j} \).

Suppose that \( t_s = t_{s-1}, t_s(\tau_i^{s}) = m > 0 \). Then there exists a \( t_s \)-valid \( w \) \( \sqsupseteq w_{s-1} \) and \( n \in \mathbb{N} \) such that one of the following holds:

(i) \( 0^n \in C_m \) for all \( v \sqsupseteq w \) and \( M_i(F_i(0^n)) \) definitely rejects.

(ii) \( 0^n \notin C_m \) for all \( v \sqsupseteq w \) and \( M_i(F_i(0^n)) \) definitely accepts.

Lastly, handling task \( r_{i,j} \) is also possible. For this we use techniques similar to previous theorems. In this setting, it is in fact possible to explicitly construct a suitable extension for the task. Due to many additional technical details required, we omit the proof of this theorem and refer to the full version of the paper.

\( \blacktriangle \)

**Theorem 8.** Let \( s \in \mathbb{N}^+ \), \( (w_{s-1}, t_{s-1}) \) defined. Consider task \( r_{i,j} \).

Suppose that \( t_s = t_{s-1}, t_s(\tau_i^{s}) = m > 0 \). Then there exists a \( t_s \)-valid \( w \) \( \sqsupseteq w_{s-1} \) and \( n \in \mathbb{N} \) such that one of the following holds:

(i) \( 0^n \in D_m \) for all \( v \sqsupseteq w \) and \( M_i(F_i(0^n)) \) definitely accepts.

(ii) \( 0^n \notin D_m \) for all \( v \sqsupseteq w \) and \( M_i(F_i(0^n)) \) definitely accepts.

We have now completed the proofs showing that the oracle construction can be performed as desired. The following theorem confirms the desired properties of \( O := \bigcup_{i \in \mathbb{N}} w_i \). Remember that \( |w_0| < |w_1| < \ldots \) is unbounded, hence for any \( z \) there is a sufficiently large \( s \) such that \( |w_s| > z \). Also remember that \( w_j \) is \( t_s \)-valid for all \( s \in \mathbb{N} \). Using these facts and the properties V1–V8 of the \( t_s \)-valid oracles, one can easily state the following result for the final oracle.

\( \blacktriangle \)

**Theorem 9.** Relative to \( O := \bigcup_{i \in \mathbb{N}} w_i \), the following holds:

(i) \( \text{NP} \cap \text{coNP} = \text{P} \), which implies \( \neg \text{NP} \cap \text{coNP} \).

(ii) No pair in \( \text{DisjNP} \) is \( \leq_m^{\text{NP}} \)-hard for \( \text{DisjUP} \), which implies \( \text{DisjNP} \).

(iii) No language in \( \text{UP} \) is \( \leq_m^{\text{NP}} \)-complete for \( \text{UP} \), i.e., \( \text{UP} \).

(iv) No pair in \( \text{DisjCoNP} \) is \( \leq_m^{\text{NP}} \)-hard for \( \text{DisjCoUP} \), which implies \( \text{DisjCoNP} \).
From Theorem 9 and known relativizable results, we obtain the following additional properties that hold relative to the oracle. See, e.g., the work of Fenner et al. [15] for a definition of the mentioned function classes $NPSV$, $NPbV$, $NP^kV$, $NPMV$ and their total variants $NPSV_t$, $NPbV_t$, $NP^kV_t$, $NPMV_t$. Here, $NEE$ means $\text{NTIME}(2^{O(2^n)})$.

**Corollary 10.** The following holds relative to the oracle $O$ constructed in this section.

(i) $P = \text{NP} \cap \text{coNP} \subsetneq \text{UP} \subsetneq \text{NP}$

(ii) $\text{UP}$, $\text{NP}$, $\text{NE}$, and $\text{NEE}$ are not closed under complement.

(iii) $\text{UP} \not\subseteq \text{coNP}$

(iv) $\text{NEE} \cap \text{TALLY} \not\subseteq \text{coNEE}$

(v) $\text{NPSV}_t \subseteq \text{PF}$

(vi) $\text{NPbV}_t \not\subseteq \text{cNPSV}_t$

(vii) $\text{NP}^kV_t \not\subseteq \text{cNPSV}_t$ for all $k \geq 2$

(viii) $\text{NPMV}_t \not\subseteq \text{cNPSV}_t$

(ix) $\text{TFNP} \not\subseteq \text{cPF}$

(x) $\text{NP} \cap \text{coNP}$ has $\leq_{\text{PM}}$-complete sets, i.e., $\neg\text{NP} \cap \text{coNP}$.

(xi) $\text{UP}$ has no $\leq_{\text{PM}}$-complete sets, i.e., $\text{UP}$.

(xii) $\text{DisjNP}$ has no $\leq_{\text{PM}}$-complete pairs, i.e., $\text{DisjNP}$.

(xiii) $\text{DisjCoNP}$ has no $\leq_{\text{PM}}$-complete pairs, i.e., $\text{DisjCoNP}$.

(xiv) No pair in $\text{DisjNP}$ is $\leq_{\text{PM}}$-hard for $\text{DisjUP}$.

(xv) No pair in $\text{DisjCoNP}$ is $\leq_{\text{PM}}$-hard for $\text{DisjCoUP}$.

(xvi) There are no $p$-optimal proof systems for $\text{TAUT}$, i.e., $\text{CON}$.

(xvii) There are no optimal proof systems for $\text{TAUT}$.

(xviii) There are no $p$-optimal proof systems for $\text{SAT}$, i.e., $\text{SAT}$.

(xx) $\text{TFNP}$ has no $\leq_{\text{PM}}$-complete problems, i.e., $\text{TFNP}$.

(xxi) $\text{NPMV}_t$ has no $\leq_{\text{PM}}$-complete functions.

(xxii) $\text{NP}$ and $\text{coNP}$ do not have the shrinking property. [7, 16]

(xxiii) $\text{NP}$ and $\text{coNP}$ do not have the separation property. [16]

(xxiv) $\text{DisjNP}$ and $\text{DisjCoNP}$ contain $\text{P}$-inseparable pairs.

**References**


Oracle with \( P = \text{NP} \cap \text{coNP} \) but No Completeness in UP, DisjNP, DisjCoNP


23 Kannan, 1979. Sipser [36] cites an unpublished work by Kannan for asking if there is a set complete for NP \( \cap \) coNP.


