


An Oracle with no UP-Complete Sets, but $NP = PSPACE$

David Dingel  

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 $NP = PSPACE$, but UP has no many-one complete sets. This combines the properties of an oracle by Hartmanis and Hemachandra [25] and one by Ogiwara and Hemachandra [38].

The oracle provides new separations of classical conjectures on optimal proof systems and complete sets in promise classes. This answers several questions by Pudlák [43], e.g., the implications $UP \implies CON^N$ and $SAT \implies TFNP$ are false relative to our oracle.

Moreover, the oracle demonstrates that, in principle, it is possible that TFNP-complete problems exist, while at the same time SAT has no p-optimal proof systems.

2012 ACM Subject Classification Theory of computation \rightarrow Problems, reductions and completeness; Theory of computation \rightarrow Oracles and decision trees; Theory of computation \rightarrow Complexity theory and logic

Keywords and phrases Computational Complexity, Promise Classes, Complete Sets, Oracle Construction

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.50

Funding *Fabian Egidy*: supported by the German Academic Scholarship Foundation.

Acknowledgements We thank the referees for their helpful feedback. Special thanks to the anonymous reviewer who pointed us to related literature and brought several open questions to our attention.

1 Introduction

We investigate relationships between the following classical conjectures on NP, optimal proof systems, and complete sets in promise classes. They are deeply rooted in formal logic, but also have far reaching practical implications. Each of these conjectures remains open.

- $P \neq NP$: P does not equal NP [12, 35]
- $NP \neq \text{coNP}$: NP does not equal coNP [16]
- CON: p-optimal proof systems for TAUT do not exist [34]
- CON^N : optimal proof systems for TAUT do not exist [34]
- SAT: p-optimal proof systems for SAT do not exist [34]
- TFNP: TFNP does not contain many-one complete problems [36]
- $NP \cap \text{coNP}$: $NP \cap \text{coNP}$ does not contain many-one complete problems [30]
- UP: UP does not contain many-one complete problems [25]
- DisjNP: DisjNP does not contain many-one complete pairs [45]
- DisjCoNP: DisjCoNP does not contain many-one complete pairs [37, 42]

We refer to Pudlák [41] for a comprehensive elaboration of the conjectures and their context.



© David Dingel, Fabian Egidy, and Christian Glaßer;
licensed under Creative Commons License CC-BY 4.0

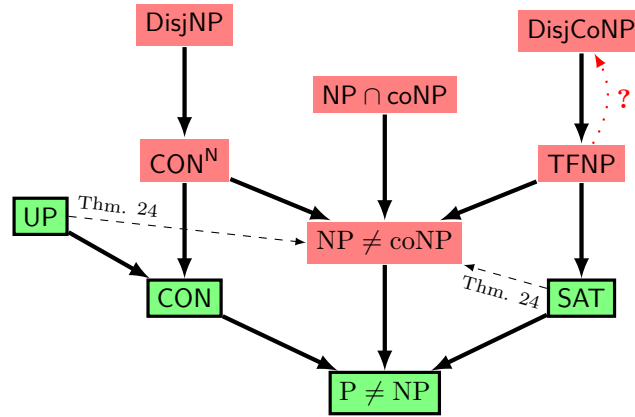
49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024).

Editors: Rastislav Královic and Antonín Kučera; Article No. 50; pp. 50:1–50:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Hypotheses that are true relative to our oracle are filled green and have borders, whereas those that are false relative to the oracle are red without borders. Solid arrows mean relativizable implications. A dashed arrow from one conjecture A to another conjecture B means that there is an oracle X against the implication $A \Rightarrow B$, meaning that $A \wedge \neg B$ holds relative to X . For clarity, we only depict the two strongest separations proved in this paper, and omit those that either follow from these, as well as those already known before. All possible separations have now been achieved, except for the one depicted as the red dotted arrow.

Pudlák’s program. The known implications between these conjectures are shown in Figure 1. They raise the question of whether further implications can be recognized with the currently available methods. Pudlák [43] therefore initiated a research program to either prove further implications or to disprove them relative to an oracle. The latter shows that the implication is beyond the reach of current methods. Hence, from today’s perspective, the corresponding conjectures are not mere reformulations of one another, but instead deserve individual attention.

So far several implications have been disproved relative to an oracle. The following selection of oracles covers all previously known disproofs of implications, in the sense that each such implication fails relative to at least one of these oracles:

- $CON^N \wedge \neg DisjNP$ (Glaßer, Selman, Sengupta, Zhang [20])
- $DisjCoNP \wedge \neg CON$ (Khaniki [31])
- $DisjNP \wedge NP \cap coNP \wedge \neg UP$ (Dose, Glaßer [15])
- $NP \cap coNP \wedge \neg CON$ (Dose [13])
- $P \neq NP \wedge \neg CON \wedge \neg SAT$ (Dose [13])
- $DisjNP \wedge UP \wedge NP \cap coNP \wedge \neg SAT$ (Dose [14])
- $DisjNP \wedge UP \wedge DisjCoNP \wedge \neg NP \cap coNP$ (Egidy, Ehrmanntraut, Glaßer [17])

Optimal proof systems. The hypotheses CON , CON^N , and SAT are concerned with the existence of (p-)optimal proof systems. The study of proof systems was initiated by Cook and Reckhow [11] and motivated by the $NP \stackrel{?}{=} coNP$ question, because the set of tautologies $TAUT$ has a proof system with polynomially bounded proofs if and only if $NP = coNP$. Krajíček and Pudlák [34, 33, 44] link questions about proof systems to questions in bounded arithmetic. Especially the notions of optimal and p-optimal proof systems have gained much attention in research. They are closely connected to the separation of fundamental complexity classes, e.g., Köbler, Messner, and Torán [32] show that the absence of optimal proof systems for $TAUT$ implies $NEE \neq coNEE$. Moreover, (p-)optimal proof systems have

tight connections to promise classes, e.g., if TAUT has p-optimal proof systems, then UP has many-one complete sets [32]. Many more relationships are known between proof systems and promise classes [3, 4, 5, 6, 7, 8, 21, 22, 32, 43, 45].

TFNP-complete problems. The class TFNP contains a multitude of problems that are of central importance to various practical applications, and as such is currently being researched with great intensity. The search for a complete problem in TFNP is of primary concern to certain fields such as cryptography, but it remains open whether such a problem exists.

For instance, the security of RSA [46] is primarily based on the hardness of integer factorization, which is a TFNP problem. However, no solid argument is known that factorization actually is a particularly hard problem within TFNP, i.e., being able to solve factorization might not require being able to solve arbitrary TFNP problems. RSA is now widely considered to no longer represent the state of the art in modern cryptography. Hence researchers develop cryptoschemes whose security is based on the hardness of other problems. However, it is still not clear whether these are strictly harder than factorization. A TFNP-complete problem would allow cryptoschemes that are at least as hard to break as any TFNP problem, i.e., schemes with optimal security guarantees.

Since the search for TFNP-complete problems has been unsuccessful, various subclasses were defined and studied in order to construct at least a complete problem for those subclasses. Their definitions are based on the proof of totality used to show the membership to TFNP. Prominent examples are PLS [29], based on the argument that *every dag has a sink*, PPA [39], based on the fact that *every graph has an even number of nodes with odd degree*, PPAD [39] and PPADS, based on slight variations of PPA for directed graphs, and PPP [40], based on the polynomial pigeonhole principle.¹ All of them have complete problems [29, 40] and can be defined in a syntactical way too [40]. Beame et al. [2] constructed oracles relative to which these classes are not a subset of P, and no inclusions exist beyond the few that are already known. The classes are significant to various practical applications like polynomially hard one way functions [9, 19], collision resistant hash functions [23], computing square roots modulo n [28], and the security of prominent homomorphic encryption schemes [27].

Our Contribution

1. Oracle with $\text{NP} = \text{PSPACE}$, but UP has no many-one complete sets.

Hartmanis and Hemachandra [25] construct an oracle relative to which UP does not have many-one complete sets. Ogiwara and Hemachandra [38] construct an oracle relative to which $\text{UP} \neq \text{NP} = \text{PSPACE}$. Our oracle provides both properties at the same time, but this is achieved using completely different methods. Due to the far-reaching collapse $\text{NP} = \text{PSPACE}$ and the close connection between UP-completeness and optimality of proof systems, we obtain a number of useful properties summarized in Corollary 25.

2. Consequences for Pudlák's program.

Regarding the conjectures shown in Figure 1, all known implications are presented in the figure, while for some of the remaining implications there exist oracles relative to which the implications do not hold. From the implications that were left open, our oracle refutes all but one. The key to this observation was to take the conjecture $\text{NP} \neq \text{coNP}$ into consideration. As we make sure that our oracle satisfies $\text{NP} = \text{coNP}$, the conjectures CON^{N} and TFNP are

¹ For precise definitions we refer to recent papers [23], [24, notably Figure 1].

false, while the conjectures CON and SAT are equivalent. In addition, the oracle satisfies the conjecture UP, which implies CON, and thereby in summary refutes all of the following previously open implications:

$$\begin{array}{lll} \text{UP} \implies \text{CON}^{\text{N}} & \text{UP} \implies \text{NP} \neq \text{coNP} & \text{UP} \implies \text{DisjNP} \\ \text{CON} \implies \text{CON}^{\text{N}} & \text{CON} \implies \text{NP} \neq \text{coNP} & \text{SAT} \implies \text{DisjCoNP} \\ \text{SAT} \implies \text{TFNP} & \text{SAT} \implies \text{NP} \neq \text{coNP} & \end{array}$$

3. Consequences for TFNP-complete problems.

Regarding the search for TFNP-complete problems, our oracle demonstrates that, in principle, it is possible that $\text{NP} = \text{coNP}$ and hence TFNP-complete problems exist, while at the same time SAT has no p-optimal proof systems.

Open questions

Hemaspaandra, Jain, and Vereshchagin [26] improve the oracle of Hartmanis and Hemachandra [25]. They show the existence of an oracle relative to which FewP [1] does not have Turing-hard sets for UP [26, Thm. 3.1]. They note that the theorem still holds when replacing the class FewP by Few [10]. Since $\text{UP} \subseteq \text{FewP} \subseteq \text{Few}$ and many-one reducibility implies Turing reducibility, this raises the following questions (in ascending order of difficulty):

Does there exist an oracle relative to which

1. $\text{NP} = \text{PSPACE}$ and UP has no set that is Turing-hard for UP?
2. $\text{NP} = \text{PSPACE}$ and FewP has no set that is Turing-hard for UP?
3. $\text{NP} = \text{PSPACE}$ and Few has no set that is Turing-hard for UP?

Regarding Pudlák’s research program, all implications that are not presented in Figure 1 fail relative to some oracle, except for one single implication, which we leave as open question:

$$\text{TFNP} \stackrel{?}{\implies} \text{DisjCoNP}$$

We suspect that such construction of an oracle with TFNP and $\neg\text{DisjCoNP}$ is a particular challenge.

2 Preliminaries

2.1 Basic Definitions and Notations

This section covers the notation of this paper and introduces well-known complexity theoretic notions.

Words and sets. All sets and machines in this paper are defined over the alphabet $\Sigma = \{0, 1\}$. Σ^* denotes the set of all strings of finite length, and for a string $w \in \Sigma^*$ let $|w|$ denote the length of w . We write \mathbb{N} for the set of non-negative integers, \mathbb{N}^+ for the set of positive integers, and \emptyset for the empty set. For a set A and $k \in \mathbb{N}$ we write $A^{=k} := \{x \in A \mid |x| = k\}$ as the subset of A containing only words of length k . We define this analogously for \leq . For a clearer notation, we use the abbreviations $\Sigma^k := \Sigma^{*=k}$ and $\Sigma^{\leq k} := \Sigma^{*\leq k}$. Let $<_{\text{lex}}$ denote the quasi-lexicographic (i.e., “shortlex”) ordering of words over Σ^* .

Functions. Let enc be a polynomial-time computable, polynomial-time invertible order-isomorphism between $(\Sigma^*, <_{\text{lex}})$ and $(\mathbb{N}, <)$, i.e., a variant of the dyadic representation. The domain and range of a function f are denoted by $\text{dom}(f)$ and $\text{ran}(f)$. For a function f and $A \subseteq \text{dom}(f)$, we define $f(A) := \{f(x) \mid x \in A\}$. We define certain polynomial functions $p_i: \mathbb{N} \rightarrow \mathbb{N}$ for $i \in \mathbb{N}$ by $p_i(n) := n^i + i$ for $i \in \mathbb{N}^+$, and $p_i(n) := 1$ for $i = 0$.

Let $\langle \cdot \rangle: \bigcup_{i \geq 0} (\Sigma^*)^i \rightarrow \Sigma^*$ be an injective, polynomial-time computable and polynomial-time invertible sequence encoding function² such that $|\langle u_1, \dots, u_n \rangle| = 2(|u_1| + \dots + |u_n| + n) + 1$. The sequence encoding function has the following property.

▷ **Claim 1.** Let $w, x \in \Sigma^*$. If $|w| \geq |x| + 3$, then $|w| \geq |\langle x, w \rangle|/4$.

Proof. The following calculation proves the claimed statement:

$$|\langle x, w \rangle| = 2(|x| + |w| + 2) + 1 = 2|x| + 5 + 2|w| \leq 2|w| + 2|w| = 4|w| \quad \triangleleft$$

Machines. We use the default model of a Turing machine in the deterministic as well as in the non-deterministic variant, abbreviated by DTM, resp., NTM. The language decided by a Turing machine M is denoted by $L(M)$. We use Turing transducers to compute functions. For a Turing transducer F we write $F(x) = y$ when on input x the transducer outputs y . We sometimes refer to the function computed by F as ‘the function F ’.

Complexity Classes and Reductions. The classes P, NP, coNP, FP and PSPACE denote the standard complexity classes. Furthermore, we are interested in several promise classes. Originally defined by Valiant [50], UP denotes the set of languages that can be decided by so called UP-machines, i.e., NTMs that have at most one accepting path on every input. The common polynomial-time many-one reducibility for sets $A, B \subseteq \Sigma^*$ is denoted by \leq_m^P , i.e., $A \leq_m^P B$ if there exists an $f \in \text{FP}$ such that $x \in A \Leftrightarrow f(x) \in B$. If \leq is some notion of reducibility for some class \mathcal{C} , then we call a set A \leq -complete for \mathcal{C} , if $A \in \mathcal{C}$ and for all $B \in \mathcal{C}$ the reduction $B \leq A$ holds.

Oracle-specific definitions and notations. An oracle B is a subset of Σ^* . We relativize the concept of Turing machines and Turing transducers by giving them access to a write-only oracle tape as proposed by Simon [48]³. Furthermore, we relativize complexity classes, proof systems, reducibilities and (p-)simulation by defining them over machines with oracle access, i.e., whenever a Turing machine or Turing transducer is part of a definition, we replace them by an oracle Turing machine or an oracle Turing transducer. We indicate the access to some oracle B in the superscript of the mentioned concepts, i.e., P^B , NP^B , FP^B , ... for complexity classes, M^B for a Turing machine or Turing transducer M , and $\leq_m^{\text{P}^B}$ for reducibility. We sometimes omit the oracles in the superscripts, e.g., when sketching ideas in order to convey intuition, but never in the actual proof.

Let $\{F_i\}_{i \in \mathbb{N}}$ be a standard enumeration of polynomial-time oracle Turing transducers, such that relative to any oracle B , F_i^B has running time exactly p_i and for any function $f \in \text{FP}^B$, there is some i such that F_i^B computes f . Since the functions computed by $\{F_i^B\}_{i \in \mathbb{N}}$ form exactly FP^B , we call such machines FP^B -machines. Similarly, let $\{N_i\}_{i \in \mathbb{N}}$ be

² Notice that the sequence encoding function explicitly is not surjective, so invertibility is meant in the sense that $\text{ran}(\langle \cdot \rangle) \in \text{P}$, and there is a function $f \in \text{FP}$ such that $\langle f(y)_1, \dots, f(y)_n \rangle = y$ for all $y \in \text{ran}(\langle \cdot \rangle)$.

³ When considering time-bounded computation, this machine model reflects the usual relativization of Turing machines. For space-bounded computations, the oracle tape is also subject to the space-bound.

a standard enumeration of polynomial-time non-deterministic oracle Turing machines, such that relative to any oracle B , N_i^B has running time exactly p_i on each path and for any set $L \in \text{NP}^B$, there is some i such that $L(N_i^B) = L$. Since the sets decided by $\{N_i^B\}_{i \in \mathbb{N}}$ form exactly NP^B , we call such machines NP^B -machines. If p is an encoding of a computation path, then $Q(p)$ denotes the set of oracle queries on p . For a Turing transducer F , an NTM N and some word x , let $Q(N(F(x)))$ denote the set of oracle queries of the computations $F(x)$ and $N(F(x))$. Hence, we interpret $N(F(x))$ as one computation which computes $F(x)$ first, followed by the computation N on input $F(x)$.

Relativized QBF. For any oracle B , we define the relativized problem QBF^B as the typical QBF problem as defined by Shamir [47], but the boolean formulas are extended by expressions $B(w)$ for $w \in \Sigma^*$, which evaluate to true if and only if $w \in B$. It holds that for any oracle B , QBF^B is $\leq_m^{\text{P}, B}$ -complete for PSPACE^B .

Throughout the remaining sections, M will be a polynomial-space oracle Turing machine which, given access to oracle B , accepts QBF^B . Choose $k \geq 3$ such that M on input $x \in \Sigma^*$ completes using at most $t(x) := |x|^k + k$ space (including queries to the oracle).

2.2 Notions for the Oracle Construction

In this section we define all necessary objects and properties for the oracle construction and convey their intuition.

Tools for UP. In order to achieve that UP^Φ has no complete set, we will ensure that for any set $L(N_i^\Phi) \in \text{UP}^\Phi$ a set $W_i^\Phi \in \text{UP}^\Phi$ exists such that W_i^Φ does not reduce to $L(N_i^\Phi)$, i.e., $W_i^\Phi \not\leq_m^{\text{P}, \Phi} L(N_i^\Phi)$. Here, W_i^Φ is a witness that $L(N_i^\Phi)$ is not complete for UP^Φ . For this, we injectively assign countably infinitely many levels of the oracle to each set W_i , where a level consists of certain words of the same length as follows.

► **Definition 2** (H_i).

Let $e(0) := 2$, $e(i+1) := 2^{e(i)}$ and $H_i := \{e(2^i \cdot 3^j) \mid j \in \mathbb{N}\}$ for $i \in \mathbb{N}$.

► **Corollary 3** (Properties of H_i).

- (i) For every $i \in \mathbb{N}$, the set H_i is countably infinite and a subset of the even numbers.
- (ii) $H_i \in \text{P}$ for all $i \in \mathbb{N}$.
- (iii) For $i, j \in \mathbb{N}$ with $i \neq j$, it holds that $H_i \cap H_j = \emptyset$.

► **Definition 4** (Witness language W_i^B).

For $i \in \mathbb{N}$ and an oracle B , we define the set

$$W_i^B := \{0^n \mid n \in H_i \text{ and there exists } x \in \Sigma^n \text{ with } x \in B\}$$

► **Corollary 5** (Sufficient condition for $W_i^B \in \text{UP}^B$).

For any oracle B and any $i \in \mathbb{N}$, the following implication holds:

$$|B^{=n}| \leq 1 \text{ for all } n \in H_i \implies W_i^B \in \text{UP}^B$$

Proof. Let B and i be arbitrary. Since $H_i \in \text{P}$, an NP-machine can reject on all inputs except 0^n with $n \in H_i$. On these inputs 0^n the machine can non-deterministically query all words $x \in \Sigma^n$ and accept if $x \in B$. Now, this machine accepts W_i^B , and if $|B^{=n}| \leq 1$, then it has at most one accepting path on all inputs. Hence, $W_i^B \in \text{UP}^B$. ◀

We can control the membership of W_i to UP in the oracle construction. We will never add more than one word on the levels H_i , except for when we can rule out N_i as a UP-machine.

Tools for $\text{NP} = \text{PSPACE}$. In order to achieve that $\text{NP}^\Phi = \text{PSPACE}^\Phi$, we will encode QBF^Φ into the oracle Φ , such that $\text{QBF}^\Phi \in \text{NP}^\Phi$, from which $\text{NP}^\Phi = \text{PSPACE}^\Phi$ follows.

► **Definition 6** (Coding set Z^B).

For any oracle B , we define the set

$$Z^B := \{x \in \Sigma^* \mid \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in B\}$$

► **Corollary 7.** For any oracle B , it holds that $Z^B \in \text{NP}^B$.

We will assemble the oracle Φ in such a way that $Z^\Phi = \text{QBF}^\Phi$, i.e., Z^Φ will be PSPACE^Φ -hard, and thus $\text{NP}^\Phi = \text{PSPACE}^\Phi$.

Goals of the construction.

► **Definition 8** (Desired properties of Φ).

The later constructed oracle Φ should satisfy the following properties:

P1 $\forall x \in \Sigma^* : (x \in \text{QBF}^\Phi \iff x \in Z^\Phi)$.

(Meaning: $\text{QBF}^\Phi \in \text{NP}^\Phi$.)

P2 $\forall i \in \mathbb{N}$, at least one of the following statements holds:

(I) $\exists x \in \Sigma^* : N_i^\Phi(x)$ accepts on more than one path.

(Meaning: N_i^Φ is not a UP^Φ -machine.)

(II) Both of the following statements hold:

(a) $\forall n \in H_i : |\Phi^{=n}| \leq 1$

(Meaning: $W_i^\Phi \in \text{UP}^\Phi$.)

(b) $\forall j \in \mathbb{N} \exists x \in \Sigma^* : N_i^\Phi(F_j^\Phi(x))$ accepts if and only if $x \notin W_i^\Phi$.

(Meaning: W_i^Φ does not reduce to $L(N_i^\Phi)$ via F_j^Φ .)

The following lemma shows that these properties imply our desired structural properties of complexity classes.

► **Lemma 9.** Let $\Phi \subseteq \Sigma^*$.

(i) If statement P1 is satisfied relative to Φ , then $\text{NP}^\Phi = \text{PSPACE}^\Phi$.

(ii) If statement P2 is satisfied relative to Φ , then there is no $\leq_m^{\text{P}, \Phi}$ -complete set for UP^Φ .

Proof. To (i): Recall that QBF^Φ is $\leq_m^{\text{P}, \Phi}$ -complete for PSPACE^Φ . Since property P1 holds relative to Φ , $Z^\Phi = \text{QBF}^\Phi$. From Corollary 7 we get $Z^\Phi \in \text{NP}^\Phi$ and thus $\text{NP}^\Phi = \text{PSPACE}^\Phi$.

To (ii): Assume there is a $\leq_m^{\text{P}, \Phi}$ -complete set C for UP^Φ . Let N_i^Φ be a UP^Φ -machine such that $L(N_i^\Phi) = C$. Then property P2.I cannot hold for i , because N_i would not be a UP -machine. Hence, property P2.II holds for i . By P2.II.a and Corollary 5, we get $W_i^\Phi \in \text{UP}^\Phi$. Since $L(N_i^\Phi)$ is complete, there is some $j \in \mathbb{N}$ such that for all $x \in \Sigma^*$ we have

$$x \in W_i^\Phi \iff F_j^\Phi(x) \in L(N_i^\Phi).$$

This is a contradiction to the satisfaction of property P2.II.b, because there has to be some x where this equivalence does not hold. Hence, the assumption has to be false and there is no $\leq_m^{\text{P}, \Phi}$ -complete set for UP^Φ . ◀

Stages. We choose certain stages on which we will deal with property P2.II.b. They are chosen as the range of a certain function m . We will call the values of m ‘UP-stages’, since only on these we will contribute to enforcing property P2.II.b and consequently UP^Φ . For now, it suffices to know that the stages will be chosen ‘sufficiently large’.

We choose a total injective function $m : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, $(i, j) \mapsto m(i, j)$ which meets the following requirements:

M1 $m(i, j) \in H_i$.

(Meaning: UP-stages $m(i, \cdot)$ are important to the witness-language W_i .)

M2 $2^{m(i, j)/4} > 4(p_i(p_j(m(i, j))))^2 + 2p_i(p_j(m(i, j))) + 1$.

(Meaning: The number of words of length $m(i, j)$ is far bigger than the number of words the computation $N_i(F_j(0^{m(i, j)}))$ can query.)

M3 $m(i, j) > m(i', j') \implies m(i, j) > p_{i'}(p_{j'}(m(i', j')))$.

(Meaning: The stages are sufficiently distant from another such that for no $i', j' \in \mathbb{N}$, the computation $N_{i'}(F_{j'}(0^{m(i', j')}))$ can query any word of the UP-stages following $m(i', j')$.)

► **Observation 10.** For all $k \in \text{ran}(m)$: $\text{ran}(\langle \cdot \rangle) \cap \Sigma^k = \emptyset$, because $\langle \cdot \rangle$ only maps to odd lengths. In particular, if $|x| = m(i, j)$ for some $i, j \in \mathbb{N}$, then $x \notin \text{ran}(\langle \cdot \rangle)$.

3 Oracle Construction

In this section we will construct an oracle Φ such that UP^Φ has no complete set and $\text{NP}^\Phi = \text{PSPACE}^\Phi$.

Construction of Φ . We construct the oracle Φ sequentially. For each $x \in \Sigma^*$, we decide whether to add some words to the oracle. We give a brief sketch of the construction and its ideas:

To P1: Whenever the input x has the form $\langle x', 0^{t(x')} \rangle$, we add $\langle x', w \rangle$ for some $w \in \Sigma^{t(x')}$ to the oracle if and only if $x' \in \text{QBF}$. Since $M(x')$ cannot query words of length $|\langle x', w \rangle|$, the membership of x' to QBF does not change after adding $\langle x', w \rangle$. From then on, we never add shorter words to Φ , thus the encoding persists correct for the finished oracle.

To P2: On every UP-stage $n := m(i, j)$, we try to achieve property P2.I or P2.II. For this, we differentiate between three cases.

1. If $N_i(F_j(0^n))$ accepts relative to the oracle constructed so far, we leave the stage n empty. In subsequent iterations, we ensure that added words will not interfere with this accepting path. This achieves property P2.II, because $0^n \notin W_i$ and $F_j(0^n) \in L(N_i)$.
2. If $N_i(F_j(0^n))$ rejects relative to the oracle constructed so far, we add a word of length n to the oracle such that this computation keeps rejecting. This achieves property P2.II, because $0^n \in W_i$ and $F_j(0^n) \notin L(N_i)$. Notice that finding appropriate words may not be possible.
3. If $N_i(F_j(0^n))$ rejects relative to the oracle constructed so far, and there is no choice of a word to add such that $N_i(F_j(0^n))$ keeps rejecting (i.e., case 2 is not possible), then we force $N_i(F_j(0^n))$ to accept on two different paths (which is possible, as we will show). In subsequent iterations, we ensure that added words will not interfere with these accepting paths. This achieves property P2.I.

In iterations after a UP-stage, if we have to add words to maintain P1, we choose them in such a way that property P2 still remains upheld for the previous stage. This is captured in Procedure 3, `handle_UP_stage_aftercare`. Once the next stage is reached, our word length will have increased enough such that the stage before cannot be affected anymore.

► **Definition 11** (Oracle construction).

Define $\Phi_0 := \emptyset$. For $k \in \mathbb{N}^+$, define $\Phi_{k+1} := \Phi_k \cup \text{construct}(\text{enc}(k), \Phi_k)$ according to Procedure 1. Finally, define $\Phi := \bigcup_{k \in \mathbb{N}} \Phi_k$.

■ **Procedure 1** $\text{construct}(x, B)$.

```

1 If  $x = \langle x', 0^{t(x')} \rangle$  and  $M^B(x')$  accepts:
2   If the largest UP-stage  $m(i, j) < |x|$  exists:
3     Return  $\text{handle\_UP\_stage\_aftercare}(x', i, j, B)$ 
4   Return  $\{x\}$ 
5 Elif  $x = 0^{m(i, j)}$  for some  $i, j \in \mathbb{N}$ :
6   Return  $\text{handle\_UP\_stage}(i, j, B)$ 
7 Else:
8   Return  $\emptyset$ .
```

■ **Procedure 2** $\text{handle_UP_stage}(i, j, B)$.

```

1  $n := m(i, j)$ 
2 If  $N_i^B(F_j^B(0^n))$  accepts:
3   Return  $\emptyset$ 
4 // Here,  $N_i^B(F_j^B(0^n))$  rejects
5 If  $\exists y \in \Sigma^n$  s.th.  $N_i^{B \cup \{y\}}(F_j^{B \cup \{y\}}(0^n))$  rejects:
6   Return  $\{y\}$ 
7 Else:
8   Let  $y, z \in \Sigma^n$  s.th.  $N_i^{B \cup \{y, z\}}(F_j^{B \cup \{y, z\}}(0^n))$  accepts on  $\geq 2$  paths
9   Return  $\{y, z\}$ 
```

■ **Procedure 3** $\text{handle_UP_stage_aftercare}(x', i, j, B)$.

```

1  $n := m(i, j)$ 
2 If  $N_i^B(F_j^B(0^n))$  accepts on at least one path  $p_1$ :
3    $Q := \begin{cases} Q(p_1) \cup Q(p_2) & \text{if } N_i^B(F_j^B(0^n)) \text{ accepts on another path } p_2 \neq p_1 \\ Q(p_1) & \text{else} \end{cases}$ 
4   Choose  $w \in \Sigma^{t(x')}$  s.th.  $\langle x', w \rangle \notin Q$ 
5   Return  $\{\langle x', w \rangle\}$ 
6 // Here,  $N_i^B(F_j^B(0^n))$  rejects
7 If  $\exists y \in \{\langle x', w \rangle \mid w \in \Sigma^{t(x')}\}$  s.th.  $N_i^{B \cup \{y\}}(F_j^{B \cup \{y\}}(0^n))$  rejects:
8   Return  $\{y\}$ 
9 Let  $y, z \in \{\langle x', w \rangle \mid w \in \Sigma^{t(x')}\}$  s.th.  $N_i^{B \cup \{y, z\}}(F_j^{B \cup \{y, z\}}(0^n))$  accepts on  $\geq 2$  paths
10 Return  $\{y, z\}$ 
```

Notice that in `handle_UP_stage` reaching line 3 corresponds to case 1 of the construction sketch for P2, reaching line 5 corresponds to case 2, and reaching lines 7–8 corresponds to case 3. In lines 2–4 of `construct`, because M accepts, we need to add at least one word $\langle x', w \rangle$ to the oracle in order to achieve P1. Procedure 3, `handle_UP_stage_aftercare`, ensures that a previous UP-stage either remains unaffected by this (lines 2–7), or at least i now satisfies property P2.I instead (lines 8–9).

It is not clear that the oracle construction can be performed as stated, because lines 4 and 8 in `handle_UP_stage_aftercare` and line 7 in `handle_UP_stage` claim the existence of words with complex properties. The following claims attend to this problem.

▷ **Claim 12** (Line 7 in `handle_UP_stage` can be performed).

If the line 7 is reached in `handle_UP_stage` in the step `construct(enc(k), Φ_k)`, then y and z can be chosen as stated.

50:10 An Oracle with no UP-Complete Sets, but NP = PSPACE

Proof. Let $x := \text{enc}(k)$. When reaching this line, then for some $i, j \in \mathbb{N}$ with $n := m(i, j)$ we have that $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ rejects, and for all extensions $\Phi_k \cup \{x'\}$ with $x' \in \Sigma^n$, $N_i^{\Phi_k \cup \{x'\}}(F_j^{\Phi_k \cup \{x'\}}(0^n))$ accepts. Since the accepting paths appear after adding x' to Φ_k , these paths must query x' . There are 2^n words of length n and an accepting path can query at most $p_i(p_j(n)) + p_j(n) \leq 2p_i(p_j(n))$ of these words. Let X be a set consisting of $4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) + 1$ pairwise different words x' of length n . By M2,

$$2^{n/4} > 4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) + 1,$$

whereby such a set exists. We show that there must be two different words $y, z \in X$ such that $N_i^{\Phi_k \cup \{y\}}(F_j^{\Phi_k \cup \{y\}}(0^n))$ (resp., $N_i^{\Phi_k \cup \{z\}}(F_j^{\Phi_k \cup \{z\}}(0^n))$) accepts and does not query z (resp., y) on some accepting path.

To choose y , we look at $X' \subseteq X$ consisting of $2p_i(p_j(n)) + 1$ pairwise different words. When extending Φ_k by a word from X' for each word in X' separately, the resulting leftmost accepting paths query at most

$$2p_i(p_j(n)) \cdot |X'| = 2p_i(p_j(n)) \cdot (2p_i(p_j(n)) + 1) = 4(p_i(p_j(n)))^2 + 2p_i(p_j(n))$$

different words in total. Hence, there is at least one unqueried word $y \in X$.

To choose z , consider $N_i^{\Phi_k \cup \{y\}}(F_j^{\Phi_k \cup \{y\}}(0^n))$, which on the leftmost accepting path queries at most $p_i(p_j(n)) + p_j(n) \leq 2p_i(p_j(n))$ words. Hence, there is some unqueried word $z \in X' \subseteq X$.

So, $N_i^{\Phi_k \cup \{y\}}(F_j^{\Phi_k \cup \{y\}}(0^n))$ accepts on a path which queries y but not z , and $N_i^{\Phi_k \cup \{z\}}(F_j^{\Phi_k \cup \{z\}}(0^n))$ accepts on a path which queries z but not y (because $z \in X'$). Thus, $y \neq z$, both accepting paths are different and both are preserved when extending by both y and z , resulting in at least two different accepting paths for $N_i^{\Phi_k \cup \{y, z\}}(F_j^{\Phi_k \cup \{y, z\}}(0^n))$. \triangleleft

\triangleright Claim 13 (Line 8 in `handle_UP_stage_aftercare` can be performed).

If the line 8 is reached in `handle_UP_stage_aftercare` in the step `construct(enc(k), Φ_k)`, then y and z can be chosen as stated.

Proof. Let $x := \text{enc}(k)$. When reaching this line, then for $x = \langle x', 0^{t(x')} \rangle$ and the biggest UP-stage $n := m(i, j) \leq |x|$ we have that $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ rejects, but for all extensions $\Phi_k \cup \{y\}$ with $y \in \{\langle x', w \rangle \mid w \in \Sigma^{t(x')}\} =: Y$, the computation $N_i^{\Phi_k \cup \{y\}}(F_j^{\Phi_k \cup \{y\}}(0^n))$ accepts. Since the accepting paths appear after adding y to Φ_k , these paths must query y . By Claim 1, $|t(x')| \geq |x|/4 \geq n/4$. Hence, there are $\geq 2^{n/4}$ words $w \in \Sigma^{t(x')}$, and thus $|Y| \geq 2^{n/4}$. An accepting path can query at most $p_i(p_j(n)) + p_j(n) \leq 2p_i(p_j(n))$ of these words. Let X be a set consisting of $4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) + 1$ pairwise different words $y \in Y$. By M2,

$$2^{n/4} > 4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) + 1,$$

whereby such a set exists. From here on, we can proceed exactly as in the proof of Claim 12 to find two fitting words y and z . \triangleleft

\triangleright Claim 14 (Line 4 in `handle_UP_stage_aftercare` can be performed).

If the line 4 is reached in `handle_UP_stage_aftercare` in the step `construct(enc(k), Φ_k)`, then w can be chosen accordingly.

Proof. Let $x := \text{enc}(k)$. When reaching this line, then for $x = \langle x', 0^{t(x')} \rangle$ and the biggest UP-stage $n := m(i, j) \leq |x|$ it holds that $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ accepts. In the worst case, there are at least two accepting paths. Up to two accepting paths p_1 and p_2 query at most $2p_i(p_j(n)) + 2p_j(n) \leq 4p_i(p_j(n))$ words in total. Together with M2, we get

$$|Q(p_1) \cup Q(p_2)| \leq 4p_i(p_j(n)) \leq 4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) < 2^{n/4}.$$

By Claim 1, $|t(x')| \geq |x|/4 \geq n/4$ and thus

$$|\Sigma^{t(x')}| \geq 2^{n/4} > 4(p_i(p_j(n)))^2 + 2p_i(p_j(n)) + 1 \geq |Q(p_1) \cup Q(p_2)|.$$

Consequently, there is some $w \in \Sigma^{t(x')}$ with $\langle x', w \rangle \notin (Q(p_1) \cup Q(p_2))$. \triangleleft

The Claims 12, 13, and 14 show that $\Phi_{k+1} := \Phi_k \cup \text{construct}(\text{enc}(k), \Phi_k)$ is well-defined. We make three further observations about **construct**.

► **Observation 15.** For $k \in \mathbb{N}$, **construct**($\text{enc}(k), \Phi_k$) only adds words of length $|\text{enc}(k)|$.

The next two observations follow from Observation 15 combined with Observation 10.

► **Observation 16.** Let $n \in H_i$ for some $i \in \mathbb{N}$ and $k := \text{enc}^{-1}(0^n)$.

Then only the step **construct**($0^n, \Phi_k$) can add words of length n to Φ .

► **Observation 17.** Let $x := \langle x', 0^{t(x')} \rangle$ for $x' \in \Sigma^*$ and $k := \text{enc}^{-1}(x)$.

Then only the step **construct**(x, Φ_k) can add words of the form $\langle x', w \rangle$ with $w \in \Sigma^{t(x')}$ to Φ .

Proving the Properties. Finally, we show that the properties P1 and P2 hold relative to Φ . We start with property P1.

► **Lemma 18.** The oracle Φ satisfies property P1.

Proof. First, recall that by definition of Z ,

$$x \in Z^\Phi \iff \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi \quad (1)$$

Let $x \in \Sigma^*$ be arbitrary, and consider the step of the oracle construction where Φ_k is defined as **construct**($\langle x, 0^{t(x)} \rangle, \Phi_{k-1}$). By Observation 17, only **construct**($\langle x, 0^{t(x)} \rangle, \Phi_{k-1}$) may add words of the form $\langle x, w \rangle$ for $w \in \Sigma^{t(x)}$ to Φ . From this we can draw $x \notin Z^{\Phi_{k-1}}$, and further, together with $\Phi_k \subseteq \Phi$,

$$\exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi \iff \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi_k \quad (2)$$

The lines 5 and 6 in **construct**($\langle x, 0^{t(x)} \rangle, \Phi_{k-1}$) are skipped, since, by M1, $m(i, j)$ is always even, whereas $\langle x, 0^{t(x)} \rangle$ has odd length (see Observation 10). Hence, **handle_UP_stage** will not be entered. Since at the lines 2–4 of **construct** always at least one word is added to the oracle, **construct**($\langle x, 0^{t(x)} \rangle, \Phi_{k-1}$) adds a word $\langle x, w \rangle$ with $w \in \Sigma^{t(x)}$ to Φ_{k-1} if and only if the condition in line 1 is met and these lines are entered, i.e., if $M^{\Phi_{k-1}}(x)$ accepts. This gives

$$x \in \text{QBF}^{\Phi_{k-1}} \iff \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi_k \quad (3)$$

Since $M^{\Phi_{k-1}}(x)$ has a space bound of $t(x)$, it cannot ask words longer than $t(x)$. Thereby $M^{\Phi_{k-1}}(x)$ cannot notice a transition to the oracle Φ_k , because we only add words $\langle x, w \rangle$ with $w \in \Sigma^{t(x)}$, which have length $> t(x)$. So, $M^{\Phi_k}(x)$ accepts if and only if $M^{\Phi_{k-1}}(x)$ accepts. For the same reason, $M^{\Phi_k}(x)$ cannot notice a transition to the oracle Φ , because by Observation 15, all words in the set $\Phi \setminus \Phi_k$ have length $\geq |\langle x, 0^{t(x)} \rangle|$, and thus $M^\Phi(x)$ accepts if and only if $M^{\Phi_k}(x)$ accepts. Combining these two facts gives

$$x \in \text{QBF}^\Phi \iff x \in \text{QBF}^{\Phi_{k-1}} \quad (4)$$

50:12 An Oracle with no UP-Complete Sets, but NP = PSPACE

In total, we conclude

$$\begin{aligned}
 x \in \text{QBF}^\Phi &\stackrel{(4)}{\iff} x \in \text{QBF}^{\Phi_{k-1}} \stackrel{(3)}{\iff} \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi_k \\
 &\stackrel{(2)}{\iff} \exists w \in \Sigma^{t(x)} : \langle x, w \rangle \in \Phi \\
 &\stackrel{(1)}{\iff} x \in Z^\Phi
 \end{aligned}$$

Since $x \in \Sigma^*$ was chosen arbitrarily, property P1 holds. \blacktriangleleft

Before proving Lemma 21, i.e., proving that property P2 holds, we state the helpful claim that `construct` does not alter certain accepting paths.

\triangleright **Claim 19.** Let $i, j, k \in \mathbb{N}$ such that $k > \text{enc}^{-1}(0^{m(i,j)})$, and $n := m(i, j)$. If $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ accepts (resp., accepts on more than one path), then so does $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$.

Proof. If $\text{enc}(k) \geq_{\text{lex}} 0^{p_i(p_j(n))+1}$, then by Observation 15, only words of length $> p_i(p_j(n))$ are added to Φ_k . Hence, all paths of $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ and $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ are the same, from which the claimed statement follows.

Otherwise, $\text{enc}(k) \leq_{\text{lex}} 1^{p_i(p_j(n))}$. By M3, $m(i, j)$ has to be the biggest UP-stage $\leq |\text{enc}(k)|$. Consider the step `construct`($\text{enc}(k), \Phi_k$). Either $\Phi_{k+1} = \Phi_k$ and the claim holds, or some words are added to Φ_k . The procedure `handle_UP_stage` is not entered, because $\text{enc}(k) >_{\text{lex}} 0^n$, so the condition in line 5 of `construct` is false, whereby any added words would need to be added by `handle_UP_stage_aftercare`. Further, since $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ accepts, this can only happen via the lines 3 to 5. Here, line 4 makes sure that $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ remains accepting (resp., remains accepting on more than one path), since the respective paths do not query the chosen words. \triangleleft

\blacktriangleright **Corollary 20.** Let $i, j, k \in \mathbb{N}$ such that $k > \text{enc}^{-1}(0^{m(i,j)})$, and $n := m(i, j)$. If $N_i^{\Phi_k}(F_j^{\Phi_k}(0^n))$ accepts (resp., accepts on more than one path), then so does $N_i^\Phi(F_j^\Phi(0^n))$.

\blacktriangleright **Lemma 21.** The oracle Φ satisfies property P2.

Proof. Let $i \in \mathbb{N}$ be arbitrary. Assume that for i the property P2.I does not hold. Then we show that property P2.II holds.

\triangleright **Claim 22.** The property P2.II.a holds.

Proof. Let $n \in H_i$ be arbitrary and $k := \text{enc}^{-1}(0^n)$. By Observation 16, only the step `construct`($0^n, \Phi_k$) adds words of length n to Φ . Since n has even and $\langle \cdot, \cdot \rangle$ has always odd length (see Observation 10), the condition in line 1 of `construct` evaluates to ‘false’. So, only if line 8 in `handle_UP_stage` is executed, more than one word of length n is added to Φ_k and consequently to Φ . In this case, since the sets H_0, H_1, \dots are disjoint, there is some $j \in \mathbb{N}$ with $m(i, j) = n$ where $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ accepts on more than one path. Invoking Corollary 20 with i, j and $k+1$, we get that also $N_i^\Phi(F_j^\Phi(0^n))$ accepts on more than one path. This is a contradiction to the assumption that the property P2.I does not hold for i . Hence, line 8 in `handle_UP_stage` cannot be executed during `construct`($0^n, \Phi_k$). This gives $|\Phi^{=n}| \leq 1$, as required. \triangleleft

\triangleright **Claim 23.** The property P2.II.b holds.

Proof. Let $j \in \mathbb{N}$ be arbitrary, $n := m(i, j)$, and $k := \text{enc}^{-1}(0^n)$. Consider the step `construct`($0^n, \Phi_k$). Since n has even length, the condition in line 1 of `construct` evaluates to ‘false’. But the condition in line 5 is satisfied, so `handle_UP_stage` will define which words are added to the oracle. Also recall that only at this step words of length n are added to Φ (Observation 16), i.e., $\Phi_{k+1}^{-n} = \Phi^{-n}$. We distinguish between the three cases on how Φ_{k+1} can be defined in `handle_UP_stage`.

If Φ_{k+1} is defined via line 3, then $\Phi^{-n} = \emptyset$ and $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ accepts. By Corollary 20, $N_i^{\Phi}(F_j^{\Phi}(0^n))$ accepts too. Hence, $N_i^{\Phi}(F_j^{\Phi}(0^n))$ accepts and $0^n \notin W_i^{\Phi}$, i.e., 0^n satisfies property P2.II.b.

If Φ_{k+1} is defined via line 8, then $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ accepts on more than one path. By Corollary 20, $N_i^{\Phi}(F_j^{\Phi}(0^n))$ also accepts on more than one path. But then, property P2.I holds for i , a contradiction to the assumption at the start of Lemma 21. Hence, this case cannot occur.

If Φ_{k+1} is defined by line 5, then $|\Phi^{-n}| = 1$ and $N_i^{\Phi_{k+1}}(F_j^{\Phi_{k+1}}(0^n))$ rejects. Let $k' > k + 1$ be the smallest number such that $N_i^{\Phi_{k'}}(F_j^{\Phi_{k'}}(0^n))$ accepts. Either k' does not exist and hence, $N_i^{\Phi}(F_j^{\Phi}(0^n))$ rejects, satisfying property P2.II.b.

Otherwise $\text{enc}(k') \leq_{\text{lex}} 1^{p_i(p_j(n))}$, because $N_i^{\Phi_{k'}}(F_j^{\Phi_{k'}}(0^n))$ can query only words of length $\leq p_i(p_j(n))$, and for bigger k' , only words of length $> p_i(p_j(n))$ are added (Observation 15). Consider the step `construct`($\text{enc}(k' - 1), \Phi_{k'-1}$). By M3, $m(i, j)$ is the biggest UP-stage $\leq |\text{enc}(k' - 1)|$. Since

$$0^n = \text{enc}(k) <_{\text{lex}} \text{enc}(k + 1) \leq_{\text{lex}} \text{enc}(k' - 1),$$

the condition in line 5 is not met, thus `handle_UP_stage` is skipped. Since by the minimality of k' the computation $N_i^{\Phi_{k'-1}}(F_j^{\Phi_{k'-1}}(0^n))$ rejects, $\Phi_{k'}$ can only be defined via line 9 in `handle_UP_stage_aftercare`. But then, $N_i^{\Phi_{k'}}(F_j^{\Phi_{k'}}(0^n))$ accepts on more than one path. Invoking Corollary 20 for i, j, k' , we get that also $N_i^{\Phi}(F_j^{\Phi}(0^n))$ accepts on more than one path. Consequently, property P2.I holds for i , a contradiction to the assumption at the start of Lemma 21. Hence, this case cannot occur. \triangleleft

Since i is arbitrary, the Claims 22 and 23 show that if property P2.I does not hold, property P2.II does. Thus, property P2 holds. \blacktriangleleft

4 Conclusion

The oracle from the previous section gives the following result.

► **Theorem 24.** *There exists an oracle relative to which UP has no \leq_m^P -complete sets and $\text{NP} = \text{PSPACE}$.*

Proof. This follows from the Lemmas 9, 18, and 21. \blacktriangleleft

We summarize the properties of the oracle from Theorem 24 relating to the hypotheses of Pudlák [43] and mention a few more.

► **Corollary 25.** *Relative to the oracle from Theorem 24, all of the following hold:*

1. $\text{NP} = \text{coNP}$, i.e., $\neg\text{NP} \neq \text{coNP}$.
2. UP does not have \leq_m^P -complete sets, i.e., UP.
3. TAUT does not have p -optimal proof systems, i.e., CON.
4. SAT does not have p -optimal proof systems, i.e., SAT.
5. TFNP has a complete problem, i.e., $\neg\text{TFNP}$.

6. TAUT has optimal proof systems, i.e., $\neg\text{CON}^{\text{N}}$.
7. DisjNP has \leq_m^{PP} -complete pairs, i.e., $\neg\text{DisjNP}$.
8. DisjCoNP has \leq_m^{PP} -complete pairs, i.e., $\neg\text{DisjCoNP}$.
9. $\text{NP} \cap \text{coNP}$ has \leq_m^{P} -complete sets, i.e., $\neg\text{NP} \cap \text{coNP}$.
10. $\text{P} \subsetneq \text{UP} \subsetneq \text{NP} \cap \text{coNP} = \text{NP}$.
11. The conjecture Q does not hold [18, Def. 2].
12. The conjecture Q' does not hold [18, Def. 3].
13. $\text{TFNP} \not\subseteq_c \text{PF}$.

Proof. To 1 and 2: Follows from Theorem 24.

To 3: Follows from 2 by Köbler, Messner, and Torán [32, Cor. 4.1].

To 4: Follows from 3 and 1.

To 5: Follows from 1 by Megiddo and Papadimitrou [36, Thm. 2.1].

To 6: Follows from 1 and the fact that NP has optimal proof systems [37, Thm. 3.1].

To 7: Follows from 6 by Köbler, Messner, Torán [32, Cor. 6.1].

To 8: Follows from 7 and 1.

To 9: Follows from 1 and that NP^{Φ} has \leq_m^{P} -complete sets.

To 10: Follows from $\text{P} \subseteq \text{UP} \subseteq \text{NP}$, P has complete sets, UP^{Φ} does not have complete sets (2), NP has complete sets and $\text{NP}^{\Phi} = \text{coNP}^{\Phi}$ (1).

To 11 and 12: Follows from $\text{P} \neq \text{NP} \cap \text{coNP}$ (which is implied by 10), as shown by Fenner et al. [18, Thm. 2, Prop. 9].

To 13: Follows from 11, as shown by Fenner et al. [18, Prop. 7]. ◀

References

- 1 E. W. Allender and R. S. Rubinfeld. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.
- 2 P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 303–314, New York, NY, USA, 1995. Association for Computing Machinery. doi:10.1145/225058.225147.
- 3 O. Beyersdorff. Representable disjoint NP-pairs. In *Proceedings 24th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 3328 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2004. doi:10.1007/978-3-540-30538-5_11.
- 4 O. Beyersdorff. Disjoint NP-pairs from propositional proof systems. In *Proceedings of Third International Conference on Theory and Applications of Models of Computation*, volume 3959 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2006. doi:10.18452/15520.
- 5 O. Beyersdorff. Classes of representable disjoint NP-pairs. *Theoretical Computer Science*, 377(1-3):93–109, 2007. doi:10.1016/j.tcs.2007.02.005.
- 6 O. Beyersdorff. The deduction theorem for strong propositional proof systems. *Theory of Computing Systems*, 47(1):162–178, 2010. doi:10.1007/s00224-008-9146-6.
- 7 O. Beyersdorff, J. Köbler, and J. Messner. Nondeterministic functions and the existence of optimal proof systems. *Theoretical Computer Science*, 410(38-40):3839–3855, 2009. doi:10.1016/j.tcs.2009.05.021.
- 8 O. Beyersdorff and Z. Sadowski. Do there exist complete sets for promise classes? *Mathematical Logic Quarterly*, 57(6):535–550, 2011. doi:10.1002/malq.201010021.
- 9 N. Bitansky, O. Paneth, and A. Rosen. On the cryptographic hardness of finding a nash equilibrium. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498, 2015. doi:10.1109/FOCS.2015.94.

- 10 J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical systems theory*, 23:95–106, 1990. doi:10.1007/BF02090768.
- 11 S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979. doi:10.2307/2273702.
- 12 S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. Association for Computing Machinery. doi:10.1145/800157.805047.
- 13 T. Dose. Further oracles separating conjectures about incompleteness in the finite domain. *Theoretical Computer Science*, 847:76–94, 2020. doi:10.1016/j.tcs.2020.09.040.
- 14 T. Dose. An oracle separating conjectures about incompleteness in the finite domain. *Theoretical Computer Science*, 809:466–481, 2020. doi:10.1016/j.tcs.2020.01.003.
- 15 T. Dose and C. Glaßer. NP-completeness, proof systems, and disjoint NP-pairs. In C. Paul and M. Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPICs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.STACS.2020.9.
- 16 J. Edmonds. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1966.
- 17 A. Ehrmanntraut, F. Egidy, and C. Glaßer. Oracle with $P = NP \cap \text{coNP}$, but No Many-One Completeness in UP, DisjNP, and DisjCoNP. In S. Szeider, R. Ganian, and A. Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 45:1–45:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.MFCS.2022.45.
- 18 S. A. Fenner, L. Fortnow, A. V. Naik, and J. D. Rogers. Inverting onto functions. *Information and Computation*, 186(1):90–103, 2003. doi:10.1016/S0890-5401(03)00119-6.
- 19 S. Garg, O. Pandey, and A. Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 579–604, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
- 20 C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004. doi:10.1137/S0097539703425848.
- 21 C. Glaßer, A. L. Selman, and L. Zhang. Canonical disjoint NP-pairs of propositional proof systems. *Theoretical Computer Science*, 370:60–73, 2007. doi:10.1007/11549345_35.
- 22 C. Glaßer, A. L. Selman, and L. Zhang. The informational content of canonical disjoint NP-pairs. *International Journal of Foundations of Computer Science*, 20(3):501–522, 2009. doi:10.1007/978-3-540-73545-8_31.
- 23 P. W. Goldberg and C. H. Papadimitriou. TFNP: An update. In D. Fotakis, A. Pagourtzis, and V. Th. Paschos, editors, *Algorithms and Complexity*, pages 3–9, Cham, 2017. Springer International Publishing.
- 24 M. Göös, A. Hollender, S. Jain, G. Maystre, W. Pires, R. Robere, and R. Tao. Further Collapses in TFNP. In S. Lovett, editor, *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 33:1–33:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.CCC.2022.33.
- 25 J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.
- 26 L. A. Hemaspaandra, S. Jain, and N. K. Vereshchagin. Banishing robust turing completeness. *International Journal of Foundations of Computer Science*, 04(03):245–265, 1993. doi:10.1142/S012905419300016X.
- 27 R. Jawale, Y. T. Kalai, D. Khurana, and R. Zhang. Snargs for bounded depth computations and ppad hardness from sub-exponential lwe. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 708–721, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451055.

- 28 E. Jeřábek. Integer factoring and modular square roots. *arXiv e-prints*, page arXiv:1207.5220, July 2012. doi:10.48550/arXiv.1207.5220.
- 29 D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.
- 30 R. Kannan, 1979. Sipser [49] cites an unpublished work by Kannan for asking if there is a set complete for $\text{NP} \cap \text{coNP}$.
- 31 E. Khaniki. New relations and separations of conjectures about incompleteness in the finite domain. *The Journal of Symbolic Logic*, 87(3):912–937, 2022. doi:10.1017/js1.2021.99.
- 32 J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003. doi:10.1016/S0890-5401(03)00058-0.
- 33 J. Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995. doi:10.1017/CB09780511529948.
- 34 J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic*, 54:1063–1079, 1989. doi:10.2307/2274765.
- 35 L. A. Levin. Universal search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973. doi:https://www.mathnet.ru/ppi914.
- 36 N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.
- 37 J. Messner. *On the Simulation Order of Proof Systems*. PhD thesis, Universität Ulm, 2000.
- 38 M. Ogiwara and L.A. Hemachandra. A complexity theory for feasible closure properties. In [1991] *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 16–29, 1991. doi:10.1109/SCT.1991.160240.
- 39 C. H. Papadimitriou. On inefficient proofs of existence and complexity classes. In J. Nešetřil and M. Fiedler, editors, *Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity*, volume 51 of *Annals of Discrete Mathematics*, pages 245–250. Elsevier, 1992. doi:10.1016/S0167-5060(08)70637-X.
- 40 C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.
- 41 P. Pudlák. *Logical foundations of mathematics and computational complexity: A gentle introduction*. Springer, 2013.
- 42 P. Pudlák. On some problems in proof complexity. In O. Beyersdorff, E. A. Hirsch, J. Krajíček, and R. Santhanam, editors, *Optimal algorithms and proofs (Dagstuhl Seminar 14421)*, volume 4, pages 63–63, Dagstuhl, Germany, 2014. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/DagRep.4.10.51.
- 43 P. Pudlák. Incompleteness in the finite domain. *The Bulletin of Symbolic Logic*, 23(4):405–441, 2017.
- 44 P. Pudlák. The lengths of proofs. In S. R. Buss, editor, *Handbook of Proof Theory*, pages 547–637. Elsevier, Amsterdam, 1998.
- 45 A. Razborov. On provably disjoint NP-pairs. *BRICS Report Series*, 36, 1994. doi:10.7146/brics.v1i36.21607.
- 46 R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.
- 47 A. Shamir. $\text{IP}=\text{PSPACE}$ (interactive proof= polynomial space). In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 11–15 vol.1, 1990. doi:10.1109/FSCS.1990.89519.
- 48 I. Simon. *On Some Subrecursive Reducibilities*. PhD thesis, Stanford University, 1977.

- 49 M. Sipser. On relativization and the existence of complete sets. In *Proceedings 9th ICALP*, volume 140 of *Lecture Notes in Computer Science*, pages 523–531. Springer Verlag, 1982. doi:10.1007/BFb0012797.
- 50 L. G. Valiant. Relative complexity of checking and evaluation. *Information Processing Letters*, 5:20–23, 1976. doi:10.1016/0020-0190(76)90097-1.