Balance Problems for Integer Circuits and Separations of Relativized Conjectures on Incompleteness in Promise Classes

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

vorgelegt von

Titus Dose

aus Braunschweig

Würzburg, 2020

Eingereicht am: 23.07.2020
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Christian Glaßer
2. Gutachter: Prof. Dr. Olaf Beyersdorff
Tag der mündlichen Prüfung: 21.01.2021

JESU JUVA

Jegliches menschliche Tun lässt Raum für Verbesserungen. Manchmal ist jedoch gar nicht leicht, verbesserungsfähige Aspekte auszumachen. Dies trifft auf die umfassende, über fachliche Aspekte weit hinausreichende Betreuung meines Betreuers Christian Glaßer zu. Explizit erwähnt sei das Aufspüren und Teilen interessanter und perspektivreicher Fragen, was für die vorliegende Arbeit ein fundamentaler Beitrag war, und das Ermutigen zu und Vorleben von Hartnäckigkeit in längeren Misserfolgsphasen. Danke!

Dem Zweitgutachter, Olaf Beyersdorff, danke ich herzlich für sein großzügiges Engagement.

Moreover, I am grateful to my Prague colleague Erfan Khaniki for helpful discussions and his both friendly and valuable encouragement while proving Theorem 3.3.1.

Die namentlich mir nicht bekannten Gutachter meiner Arbeiten bei diversen Konferenzen und Journalen haben diese Arbeit signifikant verbessert. Danke!

Meinen Eltern und meinen altsprachlichen Lehrern Wilhelm-Otto Hable und Karl Wagner danke ich dafür, dass sie mich schon früh engagiert in meinen Interessen gefördert und damit den Grundstein für meinen fachlichen Werdegang mitgelegt haben.

Der Studienstiftung des deutschen Volkes sei für ein Promotionsstipendium Dank gesagt.

Ein besonderer Dank gilt zuerst und zumeist Angelika und unseren Töchtern Luisa Eliana und Clara Isabell, aber auch meinen treuen Freunden Christian D. und Lukas Z. sowie meiner Herkunftsfamilie: Ihnen allen danke ich für ihre unverzichtbare Freundschaft und Unterstützung.

Soli Deo Gloria

Contents

Chapter 1

Introduction

The first three sections of this chapter give an introduction into the topics this thesis is about. Whereas the first section introduces into computational complexity theory (complexity theory for short) in general, the next two sections introduce into the two parts this thesis consists of. These three sections are followed by a brief outline. The last two sections are of rather formal interest: The fifth section lists the publications this thesis is based on, whereas the sixth section gives information about contributions of coauthors to results presented in this thesis.

1.1 Introducing Selected Parts of Complexity Theory

What Computational Complexity Theory Is about What are the principal limits of computers (or of computations in general)? This is probably one of the most natural and fundamental questions that can be asked in the field of computer science and the question is much older than modern computers[1]. Studying this question led to many precise models of computation (e.g., Turing machines, λ-calculus, random-access machines, and counter machines), which basically all were shown to have the same computational power. This suggests that all these models precisely capture what computability means in an intuitive sense. Before these models came up and allowed a precise definition of the term "computability", mathematicians only had an informal and intuitive understanding of this term, which of course made it impossible to find answers to the question of principal limits of computations. In contrast, the precise definition of computability soon led to an (until today) incomplete, but still profound understanding of the limits of computations. An illustrative example for a problem that is undecidable (i.e., unsolvable by computers) is given below.

The nowadays probably most widely used computation model within complexity theory is the Turing machine invented by Turing in 1936 (cf. [Tur37]). There were other equivalent models available before, but the Turing machine was the first model to be accepted as a precise model of computability. The Turing machine is a model for a human who sits in front of a strip of tape divided into cells, has a pencil and an eraser, and according to a fixed and finite set of precise rules manipulates the letters on the tape. The person always is in one of finitely many possible states and follows a fixed set of rules, according to which in each computation step he/she looks at the current cell, reads the letter on that cell, and then depending on the current state and the current letter (i) replaces the letter with a new one (possibly the same), (ii) moves to either the left or the right neighbor cell, and (iii) possibly changes the current state. Despite its simplicity the model of Turing machines has the same computational power

[1] Indeed, originally and until the time the field of computability theory was arising, the term "computer" did not refer to machines, but to humans.

as modern computers, which justifies that we will use the term Turing machine and algorithm interchangeably in the following. Even more, there is a single Turing machine that can compute everything a modern computer can compute: already in 1936 (cf. [Tur37]), Turing was aware of the fact that there are universal Turing machines, i.e., a single Turing machine that —when given the code of an arbitrary Turing machine M and some input x— simulates M on input x. This observation was not only crucial for theoretical computer science, but also for the development of modern (stored-program) computers. Minsky [Min67] states that the aforementioned paper by Turing [Tur37] "contains, in essence, the invention of the modern computer and some of the programming techniques that accompanied it".

From a more practical point of view, it does not make any difference whether some problem cannot be solved algorithmically at all or whether it can be solved, but each algorithm requires billions of years even if we have all computational resources available that we can possibly have in this universe. Therefore, from this perspective it is more relevant which computational tasks can be accomplished *efficiently* and can thus be considered *feasible*. Putting it more generally, the question is: what are the computational resources (e.g., running time, memory space, and randomness) of computational tasks?

Computational complexity theory investigates this topic in a structural way. It summarizes problems in classes that are defined via upper bounds on the amount of a computational resource and investigates relations between such classes.

Some Examples Let us consider some classical computational tasks:

1. Given a natural number, is the number prime?

2. Given a natural number, what is its largest prime divisor?

3. Given a graph and a natural number k, does the graph contain a clique of size k?
   (Put more vividly, given a number of persons, all pairs of those persons that know each other, and a number k, are there k persons who pairwise know each other.)

4. Given a graph, which size has its largest clique (or one of its largest cliques in case there are more than one)?

5. Given a graph, which set of nodes forms the graph's largest clique (or one of the largest cliques in case there are more than one)?

6. Given the program of a Turing machine and a natural number x, does the machine's computation on input x stop?

The tasks are ordered in ascending difficulty. In accordance with the above remarks, when we call a task difficult, we do not mean that it is difficult to find an algorithm for it, which is subjective, but that each algorithm for solving the task requires a certain amount of computational resources, which is objective.

There is a consensus in complexity theory that tasks that can be solved in polynomial running time are considered feasible, where a computational task can be solved in polynomial time if there exists a natural number k and an algorithm that solves the task such that when given an input x of size n (denoted as $|x| = n$), then it does not require more than $n^k + k$ computation steps.

Decision and Function Problems There are different types of tasks. The tasks 1, 3, and 6 above are *decision problems*. Here instead of an algorithm computing some object, an algorithm that on every input answers "yes" or "no" is required (we will alternatively say in the following that the algorithm accepts or rejects). Mathematically, we denote such a decision problem as the set of all "yes"-instances and the computational task is to determine whether an input belongs to the set or not. Moreover, we say that some algorithm accepts a set A if it accepts on inputs $x \in A$ and rejects otherwise. The remaining of the aforementioned problems are *function problems*, i.e., an algorithm is required that on some input x computes some output object y. Generally, complexity theory rather focuses on decision problems than on function problems and so does this thesis. Although function problems often seem to be more difficult than their corresponding decision problems, in many cases they are not.

For instance, consider the problems 3, 4, and 5. Algorithms for task 5 (resp., 4) can be modified in a simple way such that they solve the task 4 (resp., 3) and only need polynomially many additional computation steps. More interestingly, the converse implications hold as well: if we are given an algorithm for task 3, then using binary search, polynomially many calls of the algorithm for task 3 suffice to determine the size of the largest clique (resp., one of the largest cliques in case there are more than one). Moreover, if we are given an algorithm for task 4, iteratively removing edges whose deletion does not decrease the size of the largest clique until no such edge remains allows us to determine a clique of largest size. In both cases, we only need polynomially many calls of the algorithms for the respective allegedly more simple tasks. Hence the best algorithms for the three tasks all have "equivalent" running time, where equivalent means that the running time of each algorithm is polynomially bounded in the running time of each of the other two algorithms.

P and NP The two most prominent classes in complexity theory are P, the class of all feasible decision problems, and NP, the class of all decision problems whose solutions are of at most polynomial length and can be verified efficiently. The name NP stands for nondeterministic polynomial time and is derived from a different, but equivalent definition of the class, that we will discuss later. The question of whether P equals NP is the most prominent and most popular open problem in theoretical computer science and also one of the most famous open problems in mathematics at all. It is widely conjectured that P does not equal NP and almost all attempts to solve the P vs NP problem aim at proving $\mathrm{P} \neq \mathrm{NP}$.

NP-Completeness Some researchers [Sip19] claim that since the time in which the question came up there has only been little progress towards proving one of the assertions $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$, namely only the development of the notion of *NP-completeness* by Cook [Coo71], who proved that SAT, the satisfiability problem for propositional formulas, is NP-complete. This means that $\mathrm{SAT} \in \mathrm{NP}$ and each problem A in NP is reducible to SAT, i.e., there is an efficiently computable translation function, called reduction, that —generally without finding answers itself— translates instances of A to instances of SAT that have the same answer. Hence the fastest algorithm for A has a running time that is polynomially bounded in the running time of the fastest algorithm for SAT. As moreover, the non-existence of efficient algorithms for SAT implies $\mathrm{P} \neq \mathrm{NP}$, it holds $\mathrm{P} = \mathrm{NP}$ if and only if there are polynomial-time algorithms for SAT. This is clear progress: no matter whether $\mathrm{P} = \mathrm{NP}$ or $\mathrm{P} \neq \mathrm{NP}$ is to be proven, it suffices to study (the existence of efficient algorithms for) some NP-complete problem. Put more vividly, we cannot fail because we considered a wrong problem.

Reconsidering the Examples Let us reconsider the above computational tasks in the light of the classes and results we have discussed afterwards. Task 6 is the most famous undecidable

problem (and thus neither in P nor in NP), i.e., there exists no algorithm that solves this task [Tur37]. The aforementioned notion of universal Turing machines is central for the proof as it allows a simple diagonal argument which yields the result. All other problems or their respective corresponding decision problems are in NP, since the solutions are short and can be efficiently verified. The question of whether problem 1 is in P was a long-standing open question and finally was positively answered by Agrawal, Kayal, and Saxena [AKS04] in a breakthrough result. The decision version of task 2 is neither known to be in P nor to be NP-complete. Thus, if it is in P, then both P = NP and P ≠ NP is still possible. Either all or none of the tasks 3, 4, and 5 have polynomial-time algorithms. Problem 3 is one of Karp's famous 21 NP-complete problems [Kar72], which were the first problems to be proven NP-complete. Hence problem 3 is in P if and only if P = NP.

Promise Problems Generally, in computational complexity theory we refer to the worst-case complexity when considering the complexity of an algorithm. In practice, however, one is seldom given random (or even worst) instances of a problem. For example, in many cases it can be guaranteed that only a proper subset of the domain of all instances occurs.

Let us give a cryptological example. There is not known any efficient algorithm that —when given an odd prime p and some element x of the multiplicative group $\mathbb{F}_p^*$ of the field of order p— determines whether x generates $\mathbb{F}_p^*$. However, when being promised that for all inputs (p, x) the prime p is a safe prime, i.e., $(p-1)/2$ is also prime, then we can efficiently test whether x generates $\mathbb{F}_p^*$[2]. This fact is for instance exploited in the Diffie–Hellman–Merkle key exchange.

This motivates the study of so-called *promise problems*. These are problems where we are given some predicate, called promise, depending on the input instance and where algorithms are only required to answer correctly on those inputs that satisfy the promise. In other words and put more intuitively, the algorithm has been promised that it is only given instances that satisfy some predicate. In general and in contrast to the above example, this predicate does not need to be easy to check, which makes the promise problem fundamentally different from deciding the subset of the original problem that consists of all elements satisfying the promise.

Promise Classes Promise classes are a central object of study in this thesis and should not be confused with promise problems. The term *promise class* is an informal term and refers to a class of computational problems characterized by machines that satisfy some property, usually expressing a certain way of computation. In general, it is undecidable whether a given machine has this property. Therefore, it is called *promise*.

A popular example is the class UP, which was defined by Valiant [Val76]. A set belongs to UP if and only if it is accepted by a nondeterministic polynomial-time Turing machine that satisfies the promise "for each input, the computation has at most one accepting path".

Promise Classes play an important role in computational complexity theory. The classes of proof systems for certain sets and of disjoint NP- or coNP-pairs, both of which we will discuss later, are also promise classes.

Canonical Complete Problems The most simple way to see that classes like P or NP contain complete problems is over canonical complete problems. For example, consider NP.

We first need to explain an alternative way to define this class. Consider some Turing machine. If we allow this machine in each computation step to split up into two paths which continue the computation in different states, then we obtain a *nondeterministic* Turing machine,

[2] If there are only finitely many Sophie Germain primes (resp., safe primes), then this holds trivially. But it also holds if —as is widely conjectured— there exist infinitely many Sophie Germain primes (resp., safe primes).

which is defined to accept if it has at least one accepting path. Now we can characterize NP as the class of all those problems L for which there exists a nondeterministic algorithm that accepts L in polynomial time, where a nondeterministic algorithm works in polynomial time if all its paths have at most polynomial length. The two mentioned variants of defining NP are equivalent for the following reasons: A nondeterministic polynomial-time algorithm can nondeterministically generate all possible potential solutions, which are of at most polynomial length, and then verify or falsify them; conversely, an accepting path of a nondeterministic polynomial-time algorithm on some input can be seen as a solution for this input: it is of at most polynomial length and efficiently verifiable.

Without going into detail, we mention that it is not difficult to construct an enumeration $M_1, M_2, \ldots$ of some nondeterministic Turing machines such that M_i has running time at most $n^i + i$, for each set A in NP there exist infinitely many $i \in \mathbb{N}^+$ such that M_i accepts A, and there is some Turing machine M that on input i and x efficiently simulates M_i on input x (this is basically obtained by using clocked machines, i.e., we let the machines count their own computation steps and let them terminate as soon as they have executed a certain number of steps). Thus the sets accepted by $M_1, M_2, \ldots$ form the class NP. Now consider the problem $C = \{(i, x, 0^t) \mid M_i \text{ accepts on input } x \text{ after at most } t \text{ steps}\}$. By the properties of the enumeration $M_1, M_2, \ldots$, this set is in NP. As, moreover, for each set $A \in \text{NP}$ there exists i such that M_i accepts A and thus $x \mapsto (i, x, 0^{|x|^i+i})$ shows that A is reducible to C, the problem C is NP-complete.

Promise Classes and Complete Problems It suggests itself to follow a similar strategy for promise classes. However, in general this does not work as in many cases we do not know suitable enumerations of machines that represent the respective classes.

For example, consider the class UP. In order to define canonical complete problems for UP in the same way as above, we need an enumeration of UP-machines with analogous properties. In particular, the machines in such an enumeration need to have at most one accepting path on every input, they need to cover the whole class UP, and there needs to exist a UP-machine that can efficiently simulate each machine in the enumeration when given its number as part of the input. It is not known whether all these properties can be realized at the same time.

So for many promise classes it is an interesting area of research to investigate whether they have complete problems. Such questions will be the central object of study in the first part of this thesis. In a recent article [Pud17] Pudlák surveys several major conjectures relevant to proof complexity most of which can —roughly speaking— be stated in the way "the promise class $\mathcal{C}$ does not have complete problems". Pudlák's motivation rather comes from proof complexity and logic and the conjectures can also be formulated in a logical way. We will discuss this later on a relatively high level. Nevertheless, it is also possible to consider them from a purely complexity-theoretical point of view, which we will do in most parts of this thesis. The larger part of this thesis is dedicated to constructing oracles that separate relativized versions of the aforementioned conjectures and thus finding answers to questions asked by Pudlák [Pud17].

Before we start presenting and discussing the conjectures, we give a brief introduction into oracles and relativizable proofs.

Oracles and Relativizable Proofs What makes the P vs NP problem so hard to solve? As for the vast majority of open problems in mathematics, it is of course possible that there is a short and easily understandable proof for $\text{P} = \text{NP}$ or $\text{P} \neq \text{NP}$ such that future generations of computer scientists will consider $\text{P} = \text{NP}$ or $\text{P} \neq \text{NP}$ as a rather simple result. But still, astonishingly, we can give precise reasons why it has been difficult for us up to now to solve the problem.

Roughly speaking, the reason is that our proof techniques provably fail for the P vs NP problem. Let us argue more precisely. An oracle is an arbitrary (possibly undecidable) set that is given to a Turing machine as a black box which the machine can ask arbitrary (many) questions to and receives the correct answers in one computation step. So when all algorithms are given an oracle, we may live in a profoundly different world. It can be shown by straightforward constructions that there are worlds in which P = NP and also worlds in which P ≠ NP.

However, except for very less results[3], all results in computational complexity theory have proofs that are based on machine simulations which can be executed the same way when all machines are given access to some oracle: the simulating machines simply ask the oracles whenever the simulated machine does this. Thus most proofs have the property that they can be easily adapted so that they also work in the presence of an arbitrary oracle. However, as there are both oracles relative to which P = NP and oracles relative to which P ≠ NP, all proofs for P = NP and all proofs for P ≠ NP do not have this property. Hence solving the P vs NP problem requires fundamentally different proof techniques.

For such reasons it has become common to construct oracles showing that certain results require techniques radically different from the usual ones. Oracle constructions are often technical and complicated elaborations, but in turn they give objective and precise reasons for the difficulty of problems. Even more, they expose the crucial point where our proof techniques fail.

1.2 The Conjectures

In this section we introduce the conjectures central for the first part of this thesis and give an overview of the results we will obtain.

The conjectures all occur in [Pud17] and their connections are investigated in that same article. Let us introduce them and the notions which they arise in. We will present these from a purely complexity-theoretical point of view and afterwards briefly discuss the perspective that Pudlák has on them. We refer to [Pud13] for many more details and much more background on the two main conjectures CON and TFNP.

Proof Systems[4] The notion of proof systems was introduced by Cook and Reckhow [CR79], who define a *proof system* f for a set A to be a total, polynomial-time computable function with range A. If A equals TAUT, the set of propositional tautologies, then f is a *propositional proof system* (pps for short). If $f(x) = y$, then we call x an f-proof for y.

A proof system f is *simulated* by a proof system g if there is a polynomial p such that for each y and each f-proof x for y there exists a g-proof for y of length at most $p(|x|)$. If additionally there exists a polynomial-time computable function that translates f-proofs into corresponding g-proofs, then f is *P-simulated* by g. We call a proof system g *(length-)optimal* (resp., *P-optimal*) if it simulates (resp., P-simulates) each proof system with the same range.

Moreover, a proof system f is said to be *polynomially bounded* if there is some polynomial p such that all x have f-proofs of length at most $p(|x|)$. It follows from the definitions that polynomially bounded proof systems are optimal.

There is a broad range of research on proof systems. In this thesis, however, we mainly focus on one aspect, namely the question of whether there are optimal (resp., P-optimal) proof

[3] e.g., PSPACE ⊆ IP and the PCP theorem

[4] Both this paragraph and the next paragraph on disjoint pairs are in parts closely oriented towards corresponding parts in the paper [DG20], which was written in cooperation with Christian Glaßer. For better readability, we do without citing this paper explicitly in the two mentioned paragraphs.

systems for SAT and TAUT (or more general, for NP- and coNP-complete[5] problems). This is reflected by the following conjectures that are crucial for this thesis.

$$\begin{aligned} \mathsf{CON}^{\mathsf{N}} &= \text{Optimal pps proof systems for TAUT do not exist.} \\ \mathsf{CON} &= \text{P-optimal pps proof systems for TAUT do not exist.} \\ \mathsf{SAT} &= \text{P-optimal proof systems for SAT do not exist.} \\ \mathsf{CON} \vee \mathsf{SAT} &= \mathsf{CON} \text{ holds or } \mathsf{SAT} \text{ holds.} \end{aligned}$$

A relativizably proven result by Köbler, Messner, and Torán shows that we obtain equivalent statements if we replace TAUT (resp., SAT) by an arbitrary other coNP-complete (resp., NP-complete) set. This observation allows us to formulate these conjectures relative to some oracle.

There is a fundamental difference between proof systems for TAUT (or any other coNP-complete set) and proof systems for SAT (or any other NP-complete sets). By definition, NP-sets always have short proofs and thus also have polynomially bounded proof systems, which are optimal as was mentioned above. For that reason, we have not formulated a conjecture $\mathsf{SAT}^{\mathsf{N}}$.

In the case of SAT, the standard proof system maps (φ, a) to φ if φ is a propositional formula and a a satisfying assignment of φ and otherwise, it maps to an arbitrary fixed satisfiable formula. Clearly it is polynomially bounded and thus optimal. A concise and comprehensible example by Pudlák [Pud17] illustrates that if the standard proof system for SAT is even P-optimal, then factoring (i.e., task 2 in the previous section) is possible in polynomial time: consider the proof system g for SAT that basically works like the standard proof system for SAT, but maps an input proposition γ_n to itself if γ_n expresses "in a natural way" that n is composite or n is prime (trivially, for each n, the proposition γ_n is satisfiable and thus in SAT). For composite n, a proof of γ_n in the standard proof system encodes a non-trivial factor of n and thus, if g is P-simulated by the standard proof system, factoring is possible in polynomial time.

The question of whether optimal or P-optimal pps exist (i.e., whether CON or $\mathsf{CON}^{\mathsf{N}}$ holds), was raised by Krajíček and Pudlák [KP89] in the context of the finite consistency problem that we discuss later and that explains the notations $\mathsf{CON}^{\mathsf{N}}$ and CON.

Krajíček and Pudlák [KP89] also prove sufficient conditions for the existence of optimal and P-optimal propositional proof systems: NE = coNE implies $\neg\mathsf{CON}^{\mathsf{N}}$ and E = NE implies $\neg\mathsf{CON}$, where E (resp., NE) is the class of all problems that are accepted by deterministic (resp., nondeterministic) Turing machines in running $2^{O(n)}$. There exists an oracle [Ver91] relative to which the converses of these implications do not hold. Köbler, Messner, and Torán [KMT03] reveal a number of connections to promise classes that we will refer to several times below and moreover, they prove implications that are similar to and stronger than the above implications by Krajíček and Pudlák: for $\mathrm{EE} \stackrel{df}{=} \mathrm{DTIME}(2^{O(2^n)})$ and $\mathrm{NEE} \stackrel{df}{=} \mathrm{NTIME}(2^{O(2^n)})$ they show that (i) $\mathrm{NEE} \cap \mathrm{TALLY} \subseteq \mathrm{coNEE}$ implies $\neg\mathsf{CON}^{\mathsf{N}}$ and (ii) $\mathrm{NEE} \cap \mathrm{TALLY} \subseteq \mathrm{EE}$ implies $\neg\mathsf{CON}$. Both these implications are wrong relative to an oracle [DG19, DG20], relative to which additionally unions of disjoint NP-complete sets are NP-complete.

Sadowski [Sad02] proves that $\neg\mathsf{CON}^{\mathsf{N}}$ holds if and only if the class of all easy subsets of TAUT is uniformly enumerable. Pudlák [Pud96, Pud17] surveys the finite consistency problem, its connection to propositional proof systems, which will be explained below, and further, related open questions including the conjectures we list and discuss in the present section. Moreover, he also draws new connections between these conjectures.

Disjoint Pairs A disjoint NP-pair (resp., coNP-pair) is a pair (A, B) of two disjoint sets $A, B \in \mathrm{NP}$ (resp., $A, B \in \mathrm{coNP}$). The standard computational task for disjoint pairs is to

[5] Unless stated differently, when speaking of reducibilities or completeness in this section, then we refer to the standard polynomial-time many-one reducibility.

separate pairs, i.e., to determine which of the sets A and B an input belongs to, where in case the input is neither in A nor in B an arbitrary answer can be given. If there is a polynomial-time algorithm for this task, then the pair is called *P-separable*. Thus in other words, a pair (A, B) is P-separable if and only if there exists some set $S \in \mathrm{P}$ with $A \subseteq S$ and $B \subseteq \overline{S}$. In correspondence with the standard computational task for disjoint pairs, the standard reducibility for disjoint pairs is defined as follows [Raz94]: (A, B) is $\leq_{\mathrm{m}}^{\mathrm{pp}}$-reducible to (C, D) if there is a total polynomial-time computable function f with $f(A) \subseteq C$ and $f(B) \subseteq D$.

The notion of disjoint pairs has its origin in public-key cryptography and characterizes promise problems [EY80, ESY84, GS88]: Given some promise problem, choose A (resp., B) to be the set of all "yes"-instances (resp., "no"-instances) of the promise problem that satisfy the promise. Then (A, B) is a disjoint pair and an algorithm separating the pair simultaneously solves the promise problem. Conversely, given a disjoint pair (A, B), consider the promise problem that consists of the elements in A and the promise is that the input is in $A \cup B$. Then again, each algorithm separating the pair also solves the promise problem.

A beautiful example for a disjoint NP-pair that non-trivially is P-separable is the Clique–Coloring pair, which is due to Pudlák [Pud03]:

$$\begin{aligned} C_0 &= \{(G, k) \mid G \text{ is a graph that has a clique of size } k\} \\ C_1 &= \{(G, k) \mid G \text{ is a graph that can be colored with } k-1 \text{ colors}\}. \end{aligned}$$

As a clique of size k cannot be colored with $k-1$ colors, (C_0, C_1) is a disjoint NP-pair. The pair is P-separable [Pud03], which can be shown using combinatorial arguments by Lovász [Lov79] and Tardos [Tar88].

The following questions are two of the most popular open problems regarding disjoint NP-pairs:

1. Are there P-inseparable disjoint NP-pairs?
2. Does DisjNP, the set of all disjoint NP-pairs, contain $\leq_{\mathrm{m}}^{\mathrm{pp}}$-complete problems?

The answer to the first question is "yes" if secure public-key cryptosystems exist [GS88]. Even, Selman, and Yacobi [EY80, ESY84] conjecture that every disjoint NP-pair has a separator that is not $\leq_{\mathrm{T}}^{\mathrm{p}}$-hard for NP, which would imply that public-key cryptosystems that are NP-hard to crack do not exist.

The statement that the answer to the second question is "no" is one of the already announced conjectures in Pudlák's article [Pud17]:

$$\mathsf{DisjNP} \quad = \quad \text{DisjNP does not contain } \leq_{\mathrm{m}}^{\mathrm{pp}}\text{-complete pairs.}$$

Analogously, it is conjectured that

$$\mathsf{DisjCoNP} \quad = \quad \text{The set of all disjoint coNP-pairs does not contain } \leq_{\mathrm{m}}^{\mathrm{pp}}\text{-complete elements.}$$

The question of whether DisjNP holds was first asked by Razborov [Raz94] in the context of propositional proof systems: Razborov defined for every pps a canonical disjoint NP-pair and showed that the pair is complete if the pps is optimal, i.e., $\mathsf{DisjNP} \Rightarrow \mathsf{CON}^{\mathsf{N}} \Rightarrow \mathsf{CON}$.

Further investigations on the connection between pps and disjoint pairs can be found in [Pud03]. In that article, it is in particular shown that the canonical pair of the resolution proof system is symmetric, i.e., $(A, B) \leq_{\mathrm{m}}^{\mathrm{pp}} (B, A)$.

Beyersdorff [Bey04, Bey06, Bey07, Bey10] investigates connections between disjoint NP-pairs and pps, and in particular studies the conjectures DisjNP and $\mathsf{CON}^{\mathsf{N}}$. To single out only one

result, he shows that under reasonable assumptions on some proof system f, the canonical pair of f is complete for the class DNPP(f) of all disjoint NP-pairs for which the disjointness is efficiently provable in the proof system f, i.e., there exist meaningful subclasses of DisjNP that do contain complete pairs [GSZ09].

Several characterizations of DisjNP are given by Glaßer, Selman, and Sengupta [GSS05]. Among these are the uniform enumerability of disjoint NP-pairs and the existence of $\leq^{\mathrm{p}}_{\mathrm{m}}$-complete functions in NPSV. Glaßer, Selman, and Zhang [GSZ07] prove that the degree structure of the class of all disjoint NP-pairs and of all canonical disjoint pairs of propositional proof systems is identical. More precisely, they show that for each disjoint NP-pair there exists some pps whose canonical pair is equivalent to the former pair. An analogous statement for canonical pairs of pps and pps does not hold as is illustrated by examples due to Pudlák [Pud03] and Beyersdorff [Bey06], which show that there are non-equivalent pps with equivalent canonical disjoint pairs.

Glaßer, Selman, and Zhang [GSZ09] draw a connection between pps, disjoint pairs, and the neither proven nor disproven hypothesis that unions of two disjoint NP-complete sets are NP-complete ($\mathrm{H}_{\mathrm{union}}$ for short). According to [DG19, DG20], for each two of the three statements $\mathsf{CON}^{\mathsf{N}}$, DisjNP, and $\mathrm{H}_{\mathrm{union}}$ and each combination of their truth values there exists an appropriate oracle, except for $\neg\mathsf{CON}^{\mathsf{N}} \wedge \mathsf{DisjNP}$, which is impossible since $\mathsf{DisjNP} \Rightarrow \mathsf{CON}^{\mathsf{N}}$ [Raz94] can be proven in a relativizable way (e.g., see [GSSZ04] for a straightforward and relativizable proof).

Total Polynomial Search Problems A total polynomial search problem (TFNP problem for short), more commonly known under the name total NP search problem, (i) is represented by a polynomial p and a binary relation $R \in \mathrm{P}$ satisfying $\forall_x \exists_y \left(|y| \leq p(|x|) \wedge (x,y) \in R\right)$ and (ii) is the following computational task: on input x compute some y with $|y| \leq p(|x|) \wedge (x,y) \in R$. In other words, a total polynomial search problem is the computational task to determine some value of a nondeterministic multivalued function with values of polynomial length that are polynomially verifiable and guaranteed to exist [MP91]

A TFNP problem is polynomially many-one reducible to another TFNP problem if the former can be solved in polynomial time being allowed to ask one query to an oracle that gives solutions to the latter [JPY88]. TFNP is the class of all total polynomial search problems.

Both the notion of total polynomial search problems and the conjecture

$$\mathsf{TFNP} \quad = \quad \text{TFNP does not contain polynomially many-one complete elements}$$

were raised by Megiddo and Papadimitriou [MP91]. Besides CON, the conjecture TFNP is the main conjecture in [Pud17].

TFNP can be alternatively defined as the class of all search problems represented by NP-machines that accept every input where the computational task is —when given an input x— to find an accepting path of the machine on input x.

We obtain a similar class, denoted as NPMV_t, when we collect all multivalued functions that are computed by NP-machines that accept each input and output some word on each accepting path. Whereas the TFNP problems have solutions that can be efficiently verified, values of NPMV_t functions might not be verifiable in polynomial time.

Although there is an essential difference in how reductions for NPMV_t functions and TFNP problems are defined, the existence of complete NPMV_t functions implies the existence of complete TFNP problems [Pud17]. Hence a result by Beyersdorff, Köbler, and Messner [BKM09] stating that the existence of P-optimal proof systems for SAT implies the existence of complete functions in NPMV_t can be exploited and —together with the former result— proves

$$\mathsf{TFNP} \Rightarrow \mathsf{SAT}.$$

Moreover, both the existence of complete NPMV_t functions and the existence of complete TFNP problems imply the existence of $\leq_{\mathrm{m}}^{\mathrm{pp}}$-complete disjoint coNP-pairs [BKM09, Pud17]. The latter implication can be also expressed as

$$\mathsf{DisjCoNP} \Rightarrow \mathsf{TFNP}.$$

No Complete Sets in UP and NP ∩ coNP Let us define the last two conjectures we will consider.

$$\begin{aligned} \mathsf{UP} &= \text{UP does not contain complete problems} \\ \mathsf{NP} \cap \mathsf{coNP} &= \text{NP} \cap \text{coNP does not contain complete problems} \end{aligned}$$

In their aforementioned article [KMT03], entitled "Optimal proof systems imply complete sets for promise classes", Köbler, Messner, and Torán prove $\mathsf{UP} \Rightarrow \mathsf{CON}$ and $\mathsf{NP} \cap \mathsf{coNP} \Rightarrow \mathsf{CON} \vee \mathsf{SAT}$. Thus besides the conjecture DisjNP, the conjecture UP is another way of strengthening the conjecture CON. Interestingly and in contrast to the situation for DisjNP, it is not known whether one of the two possible implications between UP and $\mathsf{CON}^{\mathsf{N}}$ holds. Corollary 3.2.4 shows that at least one of the two possible implications, namely $\mathsf{CON}^{\mathsf{N}} \Rightarrow \mathsf{UP}$, cannot be proven using solely relativizable proof techniques.

Known Implications Parallel to introducing the conjectures, we have already mentioned the implications between them that are known to hold. Figure 1.1 gives an overview of these.

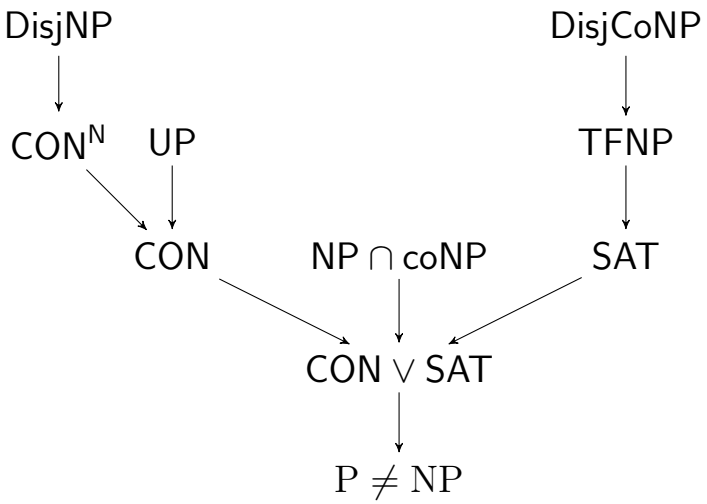


Figure 1.1: The arrows mean implications that are known to hold relative to all oracles.

For reasons of clarity, we recall the references where *relativizable* proofs of the non-trivial implications can be found.

- $\mathsf{DisjNP} \Rightarrow \mathsf{CON}^{\mathsf{N}}$: e.g., see [GSSZ04] for a straightforward, relativizable proof
- $\mathsf{UP} \Rightarrow \mathsf{CON}$: [KMT03]
- $\mathsf{NP} \cap \mathsf{coNP} \Rightarrow \mathsf{CON} \vee \mathsf{SAT}$: [KMT03]

- $\mathsf{TFNP} \Rightarrow \mathsf{SAT}$: [BKM09, Pud17]; in detail: see [BKM09] for a relativizable proof that $\neg\mathsf{SAT}$ implies the existence of complete NPMV_t functions and see [Pud17] for a relativizable proof that the existence of complete NPMV_t functions implies $\neg\mathsf{TFNP}$.

- $\mathsf{DisjCoNP} \Rightarrow \mathsf{TFNP}$: [Pud17]

As all proofs are relativizable, we can also interpret the conjectures in Figure 1.1 as the corresponding relativized statements relative to some oracle[6].

Finally, let us explain why we focus on the above selection of conjectures. The general motivation for Pudlák to investigate such conjectures is described in the next paragraph. But it has already been indicated that there are further conjectures in [Pud17] that have not been mentioned by us. The reason why we only pick the above conjectures is that Pudlák names them the "most important uniform conjectures considered in this article". To be more precise, the selection of conjectures he names that way has two slight differences from our selection above, namely:

1. As a further reference point, we also choose the popular (but of course non-uniform) conjecture CON^N, which, however, is never directly addressed in the oracles we construct.

2. We omit the conjecture RFN_1[7]. In a similar figure as Figure 1.1, Pudlák lists this conjecture between $\mathsf{CON} \vee \mathsf{SAT}$ and $\mathsf{P} \neq \mathsf{NP}$, i.e., $\mathsf{CON} \vee \mathsf{SAT} \Rightarrow \mathsf{RFN}_1 \Rightarrow \mathrm{P} \neq \mathrm{NP}$. Meanwhile, however, Khaniki [Kha19] has shown $\mathsf{CON} \vee \mathsf{SAT} \Leftrightarrow \mathsf{RFN}_1$, which removes the need of considering the conjecture separately.

An Alternative View As mentioned before, there are different views of and motivations for the conjectures we have introduced and their relations. As the first part of this thesis is part of a working program initiated by Pudlák's aforementioned article [Pud17], it inherits (part of) its motivation from Pudlák's article, which makes it inevitable to discuss Pudlák's motivation in this and also other articles.

Pudlák's article is "motivated by the problem of finding finite versions of classical incompleteness theorems" [Pud17]. Let us explain that through the example of the conjectures CON^N and CON. Both can be characterized as finite versions of incompleteness statements, more precisely as statements about some sort of finite consistency. Gödel's second incompleteness theorem, which implies his first incompleteness theorem, roughly says that each sufficiently strong theory cannot prove its own consistency. Now we can make the same step as from computability theory to complexity theory. Instead of absolute provability we can also consider efficient provability and ask the following questions: which sentences have short proofs, i.e., proofs of lengths at most polynomial in the length of the sentence? And which sentences have short proofs that can be efficiently found?

In the words of Pudlák [Pud17]: Let $\mathrm{CON}_T(n)$ for a finitely axiomatized theory T be a natural formalization of the statement "there is no derivation of contradiction of length n from the axioms of T". Krajíček and Pudlák [KP89] prove that the conjecture CON^N is equivalent to the statement that for every finitely axiomatized theory S there exists some finitely axiomatized theory T such that there exists no S-proof for $\mathrm{CON}_T(n)$ of polynomial length in n. So $\neg\mathsf{CON}^\mathsf{N}$ expresses that a very weak version of Hilbert's program (to prove the consistency of all mathematical theories) can be realized [Pud96]. Correspondingly, $\neg\mathsf{CON}$ is equivalent to the existence

[6] For the sake of simplicity, we will occasionally refer to these relativized statements as "relativized conjectures" although this is misunderstandable as these statements have only been conjectured in the unrelativized case.

[7] For a definition we refer to [Pud17].

of a theory S such that for each fixed finitely axiomatized theory T, proofs of $\mathrm{CON}_T(n)$ in S can be constructed in polynomial time in n [KP89].

Let us sketch the reason why Pudlák lays emphasis on the above conjectures (and some more that he additionally investigates in his article). All these conjectures have in common that they all say something about unprovability: The complexity-theoretical conjectures we have introduced can be equivalently formulated as statements about unprovability of certain first order sentences. So they establish a formal connection between computational complexity theory and the difficulty of proving certain sentences: high computational complexity of a problem associated with some sentence implies that the sentence is not provable in a weak theory, or requires a long proof [Pud17]. Gaining a better understanding of such connections and more fundamentally, of the general connection between logical strength of theories and computational complexity is what Pudlák is motivated by and "what the field of proof complexity basically is about" [Pud17].

Proving or disproving any of the aforementioned conjectures seems to be currently out of reach. Nevertheless, progress in finding further answers to the question of the relationships between the various conjectures may be possible by the currently available means [Pud17] and this is a large part of what Pudlák does in his article [Pud17]. He proves further implications mentioned above and is particularly interested in finding a general conjecture about incompleteness and computational complexity that contains all the above conjectures as special cases. This is his main open problem. An oracle by Khaniki [Kha19] and the oracle constructed in the proof of Theorem 3.3.1 suggest that none of the above conjectures can be such a conjecture, since proving it to be one would require non-relativizable proof techniques. We will explain this in more detail in the next paragraph.

Oracle Separations Pudlák [Pud17] does not only ask for further proofs of implications between the conjectures, but also suggests to study relativizations of the conjectures. More explicitly, he asks to "construct oracles that show that relativized conjectures are different or show they are equivalent for pairs of conjectures presented in this article" [Pud17]. For each of the oracles we construct in this thesis, there are two of the above conjectures such that the respective oracle is the first published oracle that separates the two relativized conjectures. Nevertheless, we understand Pudlák's working program in the broader sense that for each pair $\{\mathsf{A}, \mathsf{B}\}$ of conjectures both an oracle for $\mathsf{A} \not\Rightarrow \mathsf{B}$ and an oracle for $\mathsf{B} \not\Rightarrow \mathsf{A}$ is supposed to be constructed (independently of whether one of the two oracles has already been constructed).

Figure 1.2 gives an overview of all separations[8] that —to our knowledge— are published up to now. We will discuss them in the following.

Pudlák mentions that the only known separation is a separation of CON and DisjNP in [GSSZ04]. Indeed, the corresponding oracle even separates $\mathsf{CON}^{\mathsf{N}}$ and DisjNP. It is not clear whether Pudlák's working program also considers P $\neq$ NP as one of the conjectures whose relativized version is to be separated from others. On the one hand, Pudlák lists it in the figure of "the most important uniform conjectures considered in this article" [Pud17] and at least he implicitly conjectures P $\neq$ NP, since this is implied by each of the other conjectures (relative to all oracles). On the other hand, he does not explicitly introduce the conjecture P $\neq$ NP, and also does not mention an oracle by Ogiwara and Hemachandra [OH93] that separates P $\neq$ NP

[8] For the remainder of the section, when speaking of separations we always mean separations of corresponding relativized conjectures, i.e., the construction of oracles relative to which the respective conjectures are not equivalent.

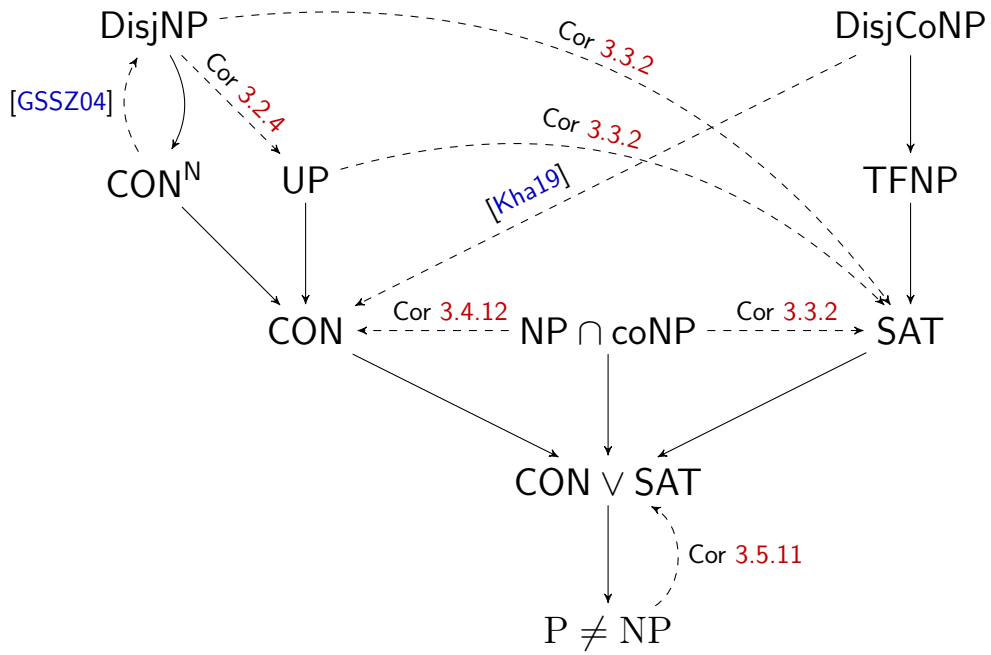


Figure 1.2: Solid arrows mean implications. A dashed arrow from one conjecture A to another conjecture B means that there is an oracle X against the implication $\mathsf{A} \Rightarrow \mathsf{B}$, which means that $\mathsf{A} \wedge \neg\mathsf{B}$ holds relative to X.

from both $\mathsf{NP} \cap \mathsf{coNP}$ and $\mathsf{CON}^{\mathsf{N}}$. In a rather straightforward oracle construction we separate the conjecture $\mathrm{P} \neq \mathrm{NP}$ from $\mathsf{CON} \vee \mathsf{SAT}$ (see Corollary 3.5.11) and by that, this conjecture has been considered exhaustively in terms of the requested oracle separations from other conjectures.

Khaniki [Kha19] is the first to address the tasks by Pudlák. He shows two of the conjectures, namely $\mathsf{CON} \vee \mathsf{SAT}$ and RFN_1, to be equivalent and constructs two oracles $\mathcal{V}$ and $\mathcal{W}$: relative to $\mathcal{V}$, there exist P-optimal propositional proof systems (i.e., $\neg\mathsf{CON}$ holds) but no many-one complete disjoint coNP-pairs (i.e., DisjCoNP holds), where —as mentioned above— the latter implies TFNP and SAT [Pud17, BKM09]. Relative to $\mathcal{W}$, there exist no optimal propositional proof systems (i.e., $\mathsf{CON}^{\mathsf{N}}$ holds), but each total polynomial search problem has a polynomial-time solution, where the latter implies $\neg\mathsf{SAT}$ relative to all oracles [KM00].Thus in particular, the relativized versions of the two main conjectures CON and TFNP are independent in the sense that neither $\mathsf{CON} \Rightarrow \mathsf{TFNP}$ nor $\mathsf{TFNP} \Rightarrow \mathsf{CON}$ holds relative to all oracles. This is not only progress in the working program introduced above, but also answers a separate question by Pudlák, who explicitly asks for such oracles that show the two main conjectures to be independent.

In this thesis we construct four oracles one of which we have already mentioned. Let us discuss the three remaining oracles and their properties.

- In Section 3.2 we construct an oracle relative to which $\mathsf{DisjNP} \wedge \mathsf{NP} \cap \mathsf{coNP} \wedge \neg\mathsf{UP}$. We later construct an oracle relative to which $\mathsf{NP} \cap \mathsf{coNP} \wedge \neg\mathsf{CON}$, which is a stronger property than $\mathsf{NP} \cap \mathsf{coNP} \wedge \neg\mathsf{UP}$. By $\mathsf{DisjNP} \wedge \neg\mathsf{UP}$, the oracle separates each of the conjectures DisjNP, $\mathsf{CON}^{\mathsf{N}}$, and CON from the conjecture UP.

- Regarding the above conjectures, one of the oracles we construct (cf. Section 3.3) extends the oracle $\mathcal{W}$ by Khaniki: relative to it, it does not only hold $\mathsf{CON}^{\mathsf{N}} \wedge \neg\mathsf{SAT}$, but even $\mathsf{DisjNP} \wedge \mathsf{UP} \wedge \mathsf{NP} \cap \mathsf{coNP} \wedge \neg\mathsf{SAT}$ (recall $\mathsf{DisjNP} \Rightarrow \mathsf{CON}^{\mathsf{N}}$ relative to all oracles)[9]. Thus it

[9]It should be mentioned that relative to $\mathcal{W}$ it does not only hold $\neg\mathsf{SAT}$, but also that all TFNP problems

proves the new separations of both (i) UP and each conjecture in {SAT, TFNP, DisjCoNP} and (ii) of NP ∩ coNP and {SAT, TFNP, DisjCoNP} and additionally, it reveals (together with Khaniki's oracle $\mathcal{V}$ [Kha19]) that DisjNP is independent of each of the conjectures SAT, TFNP, and DisjCoNP in the following sense: none of the six possible implications holds relative to all oracles.

Let us emphasize one more interesting aspect: recall that Pudlák's main open problem is to find a "general conjecture about incompleteness and computational complexity" from which the current conjectures follow as special cases [Pud17]. The aforementioned oracle $\mathcal{V}$ by Khaniki [Kha19] and our oracle show that proving one of the current conjectures to be such a general conjecture requires an unrelativizable proof: the two oracles prove that for each conjecture A there is one conjecture B that is not implied by A relative to all oracles.

Figure 1.2 illustrates that the oracle we are discussing yields one of the stronger oracle results that Pudlák [Pud17] asks for, since DisjNP, UP, and NP ∩ coNP are the strongest conjectures in their respective branches in Figure 1.2, whereas SAT is the weakest conjecture that is not implied by the three other conjectures relative to all oracles. In other words, in Figure 1.2 all conjectures on the left are known to hold and all others are known to be wrong relative to the oracle.

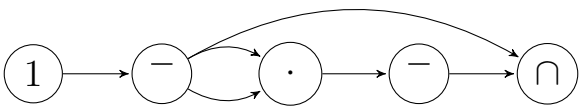
- In Section 3.4 we construct the already announced oracle relative to which NP ∩ coNP ∧ ¬CON holds. This separates each of the conjectures DisjNP, $\mathsf{CON}^{\mathsf{N}}$, and CON from NP ∩ coNP.

1.3 Integer Circuits

The History of Integer Circuits[10] Stockmeyer and Meyer [SM73] define and study membership and equivalence problems for *integer expressions*, i.e., expressions built up from single natural numbers —interpreted as singleton sets of natural numbers— by using set operations (∪, ∩, $\overline{\ }$), pairwise addition (+), and pairwise multiplication (·)[11]. For example, $\overline{\overline{1} \cdot \overline{1}} \cap \overline{1}$ describes the set of primes $\mathbb{P}$.

The *membership problem for integer expressions* asks whether some given number is contained in the set described by a given integer expression, whereas the *equivalence problem for integer expressions* asks whether two given integer expressions describe the same set. Restricting the set of allowed operations results in problems of different complexities.

Wagner [Wag84] studies a more succinct way to represent such expressions, namely *circuits over sets of natural numbers*, also called *integer circuits*. Each input gate of such a circuit is labeled with a natural number, the inner gates compute set operations or arithmetic operations ($\overline{\ }$, ∪, ∩, +, ·). The subsequent circuit computes the set of primes.



have polynomial-time solutions, which implies relative to all oracles that every optimal proof system for a set in NP is P-optimal [KM00].The latter assertion implies ¬SAT, since all non-empty sets in NP^O for arbitrary O have optimal proof systems (cf. the paragraph "Proof Systems" in the previous section).

[10] This paragraph is based on corresponding parts of the introduction of [BBD+17], which were mainly written by the author.

[11] Indeed, Stockmeyer and Meyer do not consider problems allowing pairwise multiplication in [SM73].

Starting from this circuit, one can use integer circuits to express fundamental number theoretic questions: thus a circuit describing the set of all twin primes or the set of all Sophie Germain primes can be constructed (does these circuits compute finite sets?). McKenzie and Wagner [MW07] construct a circuit C computing a set that contains 0 if and only if the Goldbach conjecture holds. By storing intermediate results in nodes and reusing them several times, we can express such questions or conjectures in a more succinct way than when using integer circuits.

Wagner [Wag84], Yang [Yan01], as well as McKenzie and Wagner [MW07] investigate the complexity of membership problems for circuits over natural numbers: here, for a given circuit C, one has to decide whether a given number n belongs to the set described by C. Travers [Tra06] and Breunig [Bre07] consider membership problems for circuits over integers and positive integers, respectively. Glaßer et al. [GHR+10] study *equivalence problems for circuits over sets of natural numbers*, i.e., the problem of deciding whether two given circuits compute the same set.

Satisfiability problems for circuits over sets of natural numbers, investigated by Glaßer et al. [GRTW10], are a generalization of the membership problems investigated by McKenzie and Wagner [MW07]: the circuits can have *unassigned input gates* and the question is: given a circuit C and a natural number b, does there exist an assignment of the unassigned input gates with natural numbers such that b is contained in the set described by the circuit?

Barth et al. [BBD+20] investigate emptiness problems for integer circuits. Here, for both circuits with unassigned inputs and circuits without unassigned inputs, the question of whether an integer circuit computes the empty set (for some/all assignment(s) if the circuits allow unassigned inputs) is raised and investigated.

Apart from the mentioned research on circuit problems there has been work on related variants like functions computed by circuits [PD09] and constraint satisfaction problems (csp) over natural numbers [GJM17, Dos16]. The constraint satisfaction problems by Glaßer, Jonsson, and Martin [GJM17] can be considered as conjunctions of equations of integer expressions with variables standing for singleton sets of natural numbers. Here the question is whether there is an assignment of the variables such that all equations are satisfied. These constraint satisfaction problems have the peculiarity that expressions describe sets of integers, whereas variables can only store singleton sets of natural numbers. The author [Dos16] addresses this and studies constraint satisfaction problems over finite subsets of $\mathbb{N}$, consequently replaces the set complement $^-$ with the set difference $-$, and allows the variables to describe arbitrary finite subsets of $\mathbb{N}$.

Our Model and Contributions The definition of the circuits investigated in this paper follows the definition of previous papers such as [MW07, GHR+10, GRTW10, BBD+20]. Yet there are some differences:

Our circuit problems are about *balanced sets* where a finite and non-empty set $S \subseteq \mathbb{N}$ is balanced if $|S| = |\{0, 1, \ldots, \max(S)\} - S|$. Analogously, S is unbalanced if $|S| \neq |\{0, 1, \ldots, \max(S)\} - S|$. That means, the maximum of a set marks the relevant area, and then we ask whether there are as many elements inside the set as outside of it.

As the notion of balanced sets only makes sense for finite sets, our circuits should solely compute finite sets. Due to that we replace the commonly used set complement $^-$ with the set difference $-$ or the symmetric difference $\triangle$.

Now, as the circuits only work over the domain of finite subsets of $\mathbb{N}$, it suggests itself to also allow the input gates of a circuit to compute arbitrary finite subsets of $\mathbb{N}$ and not only singleton sets (cf. [Dos16] where the analogous step was made for constraint satisfaction problems).

For such circuits we ask: is there an assignment of the unassigned inputs with arbitrary finite subsets of $\mathbb{N}$ under which the circuit computes a balanced set? This problem is denoted

by $\mathrm{BC}(\mathcal{O})$ where $\mathcal{O} \subseteq \{\cup, \cap, -, +, \cdot\}$ is the set of allowed operations.

The notion of balance is important in computational complexity. It occurs when considering counting classes [GNW90] like $\mathrm{C_{=}L}$ or $\mathrm{C_{=}P}$ for instance. There, the question is whether for some problem A there is a nondeterministic logarithmic-space or polynomial-time machine M accepting A, where M accepts some input x if and only if the number of accepting paths equals the number of rejecting paths.

Balance problems for integer circuits are interesting for another reason. To our knowledge, there exists neither a natural decision problem for integer circuits nor a related constraint satisfaction problem over sets of natural numbers that allows only one arithmetic operation and is known to be undecidable. In this paper, however, it is shown that $\mathrm{BC}(-, \cdot)$[12] is undecidable. Moreover, prior to this result, there were only known two problems related to integer circuits that admit no more than two operations and are known to be undecidable [GJM17, Dos16]. Both of these allow addition and multiplication.

Starting from the undecidable problem $\mathrm{BC}(-, \cdot)$, we also investigate $\mathrm{BC}(\mathcal{O})$ for arbitrary subsets of $\{-, \cdot\}$ and precisely characterize the complexity of each such problem. It turns out that all these problems are in NP. In detail, we show that $\mathrm{BC}(\cdot)$ is NL-complete, $\mathrm{BC}(-)$ is NP-complete, and $\mathrm{BC}(\emptyset) \in \mathrm{L}$.

1.4 Outline

This thesis has a simple structure. Besides the introduction and a preliminary chapter, it consists of the Chapters 3 and 4. The former deals with the construction of the four aforementioned oracles, the latter is about balance problems for integer circuits.

Chapter 3 starts with an introductory section, then contains four sections in each of which we construct one oracle, and finally ends with a brief summary. Each of the four middle sections basically consists of one fairly extensive and technical proof.

Chapter 4 consists of four sections. Between an introductory and a summarizing section, we obtain the central results in Sections 4.2 and 4.3. Section 4.2 basically consists of the proof of this chapter's main result, the undecidability of the balance problem allowing set difference and multiplication. Based on this, Section 4.3 asks the question of whether even one of this operations suffices in order to gain undecidability and answers it negatively.

1.5 Publications

This thesis contains both published results as well as unpublished results (i.e., results only published in technical reports). The former have appeared in the following refereed conference proceedings or journals.

[Dos18] T. Dose. Balance problems for integer circuits. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

[Dos19a] T. Dose. Balance problems for integer circuits. *Theor. Comput. Sci.*, 799:124–139, 2019.

[12] Consequently, $\mathrm{BC}(\triangle, \cap, \cdot)$ and $\mathrm{BC}(\triangle, \cup, \cdot)$ are undecidable as well. Both problems also allow only one arithmetic operation.

[Dos19b] T. Dose. P-optimal proof systems for each non-empty NP-set but no complete disjoint NP-pairs relative to an oracle. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

[Dos20a] T. Dose. An oracle separating conjectures about incompleteness in the finite domain. *Theor. Comput. Sci.*, 809:466–481, 2020.

[DG20] T. Dose and C. Glaßer. NP-completeness, proof systems, and disjoint NP-pairs. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

[Dos20b] T. Dose. Further oracles separating conjectures about incompleteness in the finite domain. *Theor. Comput. Sci.*, 847:76–94, 2020.

[Dos18] and [Dos19b] are the conference versions of the journal articles [Dos19a] and [Dos20a]. We recommend to ignore the conference papers and consider the journal articles instead. The technical report [DG19] is the full version of [DG20].

Let us register which parts of this thesis correspond to which of the above publications.

Section 3.2	Section 3.3	Section 3.4	Section 3.5	Chapter 4
[DG19, DG20]	[Dos19b, Dos20a]	[Dos20b]	[Dos20b]	[Dos18, Dos19a]

1.6 Contributions by Coauthors

As the list of publications and the assignment of publications to certain passages suggest, except for Section 3.2, all results within this thesis are solely due to the author of this thesis. The results in Section 3.2 were developed by Christian Glaßer in cooperation with the author and are published in [DG19, DG20]. As has been mentioned, parts of the paragraphs on proof systems and disjoint pairs in Section 1.2 are closely oriented towards these paper's introductions, which were mainly written by Christian Glaßer.

As for the sections of Chapter 3 which no coauthor has contributed to, it must be noted that these were significantly inspired by the author's collaboration with Christian Glaßer resulting in the articles [DG19, DG20]: both notations and outer structure of the proofs as well as several common proof techniques were introduced to the author by Christian Glaßer when working on the aforementioned articles.

In a further way, Christian Glaßer has contributed to this thesis by proofreading several parts and making helpful suggestions.

Chapter 2

Preliminaries

This chapter is a whole divided into three parts, one of which basic mathematical notations inhabit, elementary graph theoretic notions another one, and foundations of computational complexity theory the third.[1]

2.1 Basic Mathematical Notations

We denote the set of natural numbers and the set of integers with $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$, respectively. Moreover, we write $\mathbb{Q}$ and $\mathbb{R}$ for the set of rational and real numbers, respectively. Note $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$. $\mathbb{Z}^+$ and $\mathbb{N}^+$ denote the set of positive integers, $\mathbb{Q}^+$ is the set of positive rational numbers, and $\mathbb{R}^+$ is the set of positive real numbers. $\mathbb{Z}^-$, $\mathbb{Q}^-$, and $\mathbb{R}^-$ denote the set of negative integers, rationals, and reals, respectively. $\mathbb{P}$ is the set of primes $\{2, 3, 5, 7, 11, \dots\}$ and $\mathbb{P}_{\geq n} := \mathbb{P} \cap \{x \in \mathbb{N} \mid x \geq n\}$ for $n \in \mathbb{N}$. For $a, b \in \mathbb{Z}$ we define $[a, b]$ (resp., $[a, b)$, $(a, b]$, and (a, b)) to be the finite interval $\{x \in \mathbb{Z} \mid a \leq x \leq b\}$ (resp., $\{x \in \mathbb{Z} \mid a \leq x < b\}$, $\{x \in \mathbb{Z} \mid a < x \leq b\}$, and $\{x \in \mathbb{Z} \mid a < x < b\}$). For two integers a and b we write $a \mid b$ if a divides b, i.e., if there exists an integer c such that $b = a \cdot c$.

The Cartesian product of two sets A and B is denoted by $A \times B = \{(a, b) \mid a \in A, b \in B\}$ and the i-times Cartesian product

$$\underbrace{A \times A \times \cdots \times A}_{i \text{ times}}$$

by A^i. Furthermore, we denote by $\cup$, $\cap$, and $-$ the set operations union, intersection, and set difference, respectively, i.e., $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$, $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$, and $A - B = \{x \in A \mid x \notin B\}$. The symmetric difference is denoted by $\triangle$, i.e., $A \triangle B = (A - B) \cup (B - A)$. For the complement of a set A relative to some base set $U \supseteq A$ we write $\overline{A} = U - A$. The base set U will always be apparent from the context. $\mathcal{P}(A) := \{S \mid S \subseteq A\}$ denotes the power set of a set A and $\mathcal{P}_{\text{fin}}(A) = \{S \in \mathcal{P}(A) \mid S \text{ finite}\}$. For a finite and non-empty set A of integers let $\max(A)$ (resp., $\min(A)$) denote the maximal (resp., minimal) element of A. For a finite set A we denote the cardinality of A with $|A|$.

We extend the arithmetical operations $+$ and $\cdot$ to sets of integers: for $A, B \subseteq \mathbb{Z}$ define $A + B := \{a + b \mid a \in A, b \in B\}$ and $A \cdot B := \{a \cdot b \mid a \in A, b \in B\}$. For sets $A_1, \dots, A_n \subseteq \mathbb{Z}$ we use the notation $\prod_{i=1}^{n} A_i$ (resp., $\sum_{i=1}^{n} A_i$) for the set $A_1 \cdot A_2 \cdot \dots \cdot A_n$ (resp., $A_1 + A_2 + \dots + A_n$). Note that for $i \geq 2$ and $A \subseteq \mathbb{Z}$ the term A^i does *not* denote the set $\prod_{j=1}^{i} A$ but the i-times

[1] In the style of Gaius Iulius Caesar, Commentarii de bello Gallico, beginning of Liber I: "Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur."

Cartesian product of A. As the set $\{1\}$ is the unique neutral element regarding the multiplication of sets of integers, we let terms like $\prod_{i\in\emptyset} A$ denote the set $\{1\}$.

In this thesis, when speaking of functions we mean partial functions, i.e., a *function* f can be formally defined as a triple (A, B, G) where A, B, and G are sets and $G \subseteq A \times B$ such that for all $x \in A$ it holds $|G \cap (\{x\} \times B)| \leq 1$. Then f is a *function from* A *to* B and denoted as $f\colon A \to B$. If A and B are apparent from the context, we may omit them. If for $x \in A$ there exists some $y \in B$ with $(x, y) \in G$, then we say that $f(x)$ is *defined* and let $f(x)$ denote the unique such $y \in B$. Instead of $f(x) = y$ we may also write $x \mapsto y$ if it is apparent from the context which function we refer to. If for $x \in A$ it does not hold that $f(x)$ is defined, then we say that $f(x)$ is *undefined*.

The set $\{x \in A \mid f(x) \text{ is defined}\}$ is called the *domain of* f and denoted with $\mathrm{dom}(f)$. If $\mathrm{dom}(f) = A$, then the partial function f is called *total*. The *range* $\mathrm{ran}(f)$ of a function f is the set $\{f(x) \mid x \in A\} \subseteq B$. Let $X \subseteq A$ and $Y \subseteq B$. The *image* of X under f, denoted as $f(X)$, is defined to be the set $\{f(x) \mid x \in X\} \subseteq B$. Moreover, we denote the *preimage* of Y under f by $f^{-1}(Y)$, i.e., $f^{-1}(Y) = \{x \in A \mid f(x) \in Y\}$.

The function f is *injective* if $f(x) \neq f(y)$ for all distinct $x, y \in \mathrm{dom}(f)$, f is *onto* if $\mathrm{ran}(f) = B$, and f is *bijective* if it is total, injective and onto. The *support* $\mathrm{supp}(f)$ of a function $f\colon A \to B$ with $B \subseteq \mathbb{R}$ is the set $\{x \in \mathrm{dom}(f) \mid f(x) \neq 0\}$. We say that a partial function f is *injective on its support* if $f(x) \neq f(y)$ for all distinct $x, y \in \mathrm{supp}(f)$. If a partial function f is not defined at point x, then $f \cup \{x \mapsto y\}$ denotes the extension f' of f that at x has value y and satisfies $\mathrm{dom}(f') = \mathrm{dom}(f) \cup \{x\}$. For an injective function $f\colon A \to B$ the *inverse function* of f, denoted as $f^{-1}\colon B \to A$, is the unique function from B to A with domain $\mathrm{ran}(f)$ that satisfies $f^{-1}(f(x)) = x$ for all $x \in \mathrm{dom}(f)$.

The greatest common divisor of positive naturals x and y, i.e., the greatest positive natural number that divides both x and y, is denoted by $\gcd(x, y)$. The logarithm function log denotes the function $\mathbb{N}^+ \to \mathbb{N}$ defined by $x \mapsto \max(\{k \in \mathbb{N} \mid 2^k \leq x\})$. $\sqrt{\cdot}$ denotes the standard square root function $(\mathbb{R}^+ \cup \{0\}) \to \mathbb{R}$, i.e., for $y \in \mathbb{R}^+ \cup \{0\}$, $\sqrt{y}$ is the unique non-negative real number x with $x^2 = y$. For positive natural numbers $n \geq k$ we define $\binom{n}{k} = \prod_{i=1}^{k} \frac{n+1-i}{i}$.

We fix the alphabet $\Sigma = \{0, 1\}$ and denote the length of a word $w \in \Sigma^*$ with $|w|$. Let $\Sigma^{\prec n} = \{w \in \Sigma^* \mid |w| \prec n\}$ for $\prec \in \{\leq, <, =, >, \geq\}$. We use Σ^n as an abbreviation for $\Sigma^{=n}$. Moreover, $\Sigma^{[m,n]} = \{w \in \Sigma^* \mid |w| \in [m, n]\}$ for $m, n \in \mathbb{N}$. The empty word is denoted by ε and the i-th letter of a word w for $0 \leq i < |w|$ is denoted by $w(i)$, i.e., $w = w(0)w(1)\cdots w(|w| - 1)$. For $k \leq |w|$ let $\mathrm{pr}_k(w) = w(0)\cdots w(k-1)$. A word v is a prefix of a word w if there exists some $k \leq |w|$ such that $v = \mathrm{pr}_k(w)$. If v is a prefix of w, then we write $v \sqsubseteq w$ or $w \sqsupseteq v$. If $v \sqsubseteq w$ and $|v| < |w|$, then we write $v \sqsubsetneq w$ or $w \sqsupsetneq v$.

For each finite set $Y \subseteq \Sigma^*$, let $\ell(Y) = \sum_{w\in Y} |w|$.

The *quasi-lexicographical order* of Σ^* is the total order $\leq \subseteq \Sigma^* \times \Sigma^*$ defined by

$$u \leq v \Leftrightarrow \Big(|u| < |v| \vee \Big(|u| = |v| \wedge \exists_{0\leq i<|u|}\big(u(i) = 0 \wedge v(i) = 1 \wedge \forall_{0\leq j<i} u(j) = v(j)\big)\Big) \vee u = v\Big).$$

An *ω-word* w is a total function $\mathbb{N} \to \Sigma$.

If A is a set, then $A(x)$ denotes the *characteristic function at point* x, i.e., $A(x)$ is 1 if $x \in A$, and 0 otherwise. The *semi-characteristic function* of A is a function with domain A and range $\{1\}$. Let $w_0, w_1, \ldots$ be the elements of Σ^* in quasi-lexicographical order, i.e., $w_0 = \varepsilon$, $w_1 = 0$, $w_2 = 1$, and so on. In case A is a set of words, the *characteristic sequence* of A is defined to be the ω-word $A(w_0)A(w_1)\ldots$, i.e., the function $\mathbb{N} \to \{0, 1\}$ mapping $i \in \mathbb{N}$ to $A(w_i)$.

2.2 Graphs

In this thesis we are mainly interested in rather specific graphs, namely finite directed acyclic multigraphs satisfying the property that the indegree of every node is at most 2. Therefore, some of the following notions are defined less general than usual. For example, we define graphs to be always finite and we do not differentiate between walks, trails, and paths.

A *directed multigraph* is a pair (V, E) where V is a finite and non-empty set and E is a finite subset of $V \times V \times \mathbb{N}$. For the sake of simplicity, we require that for all directed multigraphs (V, E) and all $(u, v, i) \in E$ with $i > 0$ it holds $(u, v, i-1) \in E$. The elements of V are called *nodes*, the elements of E are called *edges*. If $E \subseteq V \times V \times \{0\}$, then the directed multigraph (V, E) is called a *directed graph*.

Consider an edge (u, v, i). The node u (resp., v) is the *source node* (resp., *target node*) of the edge (u, v, i) and the edge (u, v, i) is an *outgoing* (resp., *incoming*) edge of u (resp., v). For nodes $u, v \in V$ the node u (resp., v) is called a *direct predecessor of* v (resp., *direct successor of* u) if $(u, v, 0) \in E$. Moreover, u (resp., v) is called a *predecessor of* v (resp., *successor of* u) if u is a direct predecessor of v or u is a predecessor of some direct predecessor of v.

Let us define the notion of *paths*. Let (V, E) be a directed multigraph. A *path of length* n for $n \in \mathbb{N}$ is an element $((u_1, v_1, i_1), \ldots, (u_n, v_n, i_n))$ of E^n such that for all $j = 1, \ldots, n-1$ it holds $v_j = u_{j+1}$. This path is called a *path from* u_1 *to* v_n. Note that in particular, $()$ is a path of length 0 from u to u for all $u \in V$. A path $((u_1, v_1, i_1), \ldots, (u_n, v_n, i_n))$ for $n \in \mathbb{N}^+$ is called a *cycle* if $u_1 = v_n$. A directed multigraph (V, E) is *acyclic* if there is no cycle in it.

Let (V, E) be a directed multigraph and $u \in V$. The *indegree* of the node u is defined to be $|E \cap (V \times \{u\} \times \mathbb{N})|$. The *outdegree* of u is $|E \cap (\{u\} \times V \times \mathbb{N})|$.

2.3 Computational Complexity

2.3.1 Turing Machines and Transducers

Throughout this thesis we use the standard model of Turing machines: a Turing machine has

- a read-only input tape,
- a constant number of working tapes that are infinite to both sides,
- possibly one write-only output tape with infinite space only to the right side,
- for each tape a head that points at the current cell of the respective tape,
- a finite set of states among which there are an initial state, an accepting state, and a rejecting state, as well as
- a finite set of rules that in every situation specify the next computation steps (there is not necessarily only one next computation step).

In the following we briefly sketch the operating principles of Turing machines. For formal definitions we refer to standard textbooks such as [Pap94] and [AB09].

Computations, Configurations, and Acceptance Behavior Let M be a Turing machine and $x \in \Sigma^*$ be some input of M. We denote the computation of M on input x with $M(x)$. The current situation of M is specified in a *configuration* of M, i.e., a configuration consists of the following information: (i) the current state, (ii) the contents of all tapes, and (iii) the current positions of all heads. The machine M on input x starts in the initial state with x written on

the input tape, the head of the input tape points at the first symbol of x, all other tapes are empty, and the head of the output tape (if existent) points at the first cell of the tape. We call this configuration the *initial configuration* of M on input x. If the state of a configuration is the accepting (resp., rejecting) state, then we call the configuration *accepting* (resp., *rejecting*). A configuration is called a *stop configuration* if it is accepting or rejecting.

A *computation path* of a computation $M(x)$ is a sequence of configurations with the following properties:

- The first configuration is the initial configuration of M on input x.
- Each configuration C of the sequence either (i) is a stop configuration not followed by another configuration or (ii) it is followed by a configuration C' such that M can reach C' from C in one step.

A computation path is called *accepting* (resp., *rejecting*) if it contains an *accepting* (resp., *rejecting*) configuration. It then follows that the path is finite and the last configuration is the unique stop configuration of the path. The computation $M(x)$ *accepts* if there exists at least one accepting computation path of M on input x. Otherwise, $M(x)$ *rejects*. We denote $L(M) = \{x \mid M(x) \text{ accepts}\}$. We say that a computation *terminates* if all its computation paths are finite.

If for each x there is only one computation path of M on input x, then M is called a *deterministic Turing machine*. Otherwise, M is called a *nondeterministic Turing machine*.

A set $L \subseteq \Sigma^*$ is called *recursively enumerable* or *computably enumerable* if $L(M) = L$ for some Turing machine M. If there exists a deterministic Turing machine M with $L(M) = L$ such that for each input x the computation $M(x)$ terminates, then L is called *decidable*. In that case we say that that M *decides* the language L.

Transducers A deterministic Turing machine is called a *(Turing) transducer* if it additionally has a write-only output tape (the head of the output tape always points at the first empty cell and in each step there may be added one symbol to the word currently on the output tape). Hence a transducer F computes a function and more precisely, it computes a function $f\colon \Sigma^* \to \Sigma^*$ if for all inputs x the following holds.

- If $f(x)$ is defined, then the computation of F on input x is accepting and in the (unique) accepting configuration, the content of the output tape is $f(x)$.
- If $f(x)$ is not defined, then the computation of F on input x is not accepting (i.e., the unique computation path of F on input x is either infinite or ends with a rejecting configuration).

In particular, $L(F) = \mathrm{dom}(f)$.

We identify a transducer F with the function that it computes. Therefore, depending on the context, $F(x)$ either denotes the computation of F on input x or the output of this computation.

A function f is called *computable* if there exists a Turing transducer that computes f. Note that a set $L \subseteq \Sigma^*$ is decidable if and only if its characteristic function is computable.

Time and Space Bounds Let $s, t\colon \mathbb{N} \to \mathbb{N}$ be total functions and M be an arbitrary k-tape Turing machine.

- We say that M works in space s if for all $x \in \Sigma^*$ and all configurations of computation paths of M on input x, each working tape of M contains at most $s(|x|)$ symbols. Note that we neither count the symbols on the input tape nor the symbols on the output tape (if existent).

- We say that M works in time t if for all $x \in \Sigma^*$ each computation path of M on input x consists of at most $t(|x|)$ configurations.

We say that M is a *polynomial-time Turing machine* if there exists some $i \in \mathbb{N}^+$ such that M works in time $n \mapsto n^i + i$. Analogously, we call M a *logarithmic-space Turing machine* (resp., *polynomial-space Turing machine*) if there exists some $i \in \mathbb{N}^+$ such that M works in space $n \mapsto i \cdot \log n + i$ (resp., $n \mapsto n^i + i$). A function f is called *polynomial-time computable* if there exists a polynomial-time Turing transducer that computes f. Analogously, a function f is called *logarithmic-space computable* (resp., *polynomial-space computable*) if there exists a logarithmic-space Turing transducer (resp., polynomial-space Turing transducer) that computes f. An injective function f is called *polynomial-time invertible* (resp., *logarithmic-space invertible*) if its inverse function is polynomial-time computable (resp., logarithmic-space computable).

Identification of Σ^* and $\mathbb{N}$ Throughout this thesis we identify Σ^* with $\mathbb{N}$ via the bijection $\Sigma^* \to \mathbb{N}$ given by $w \mapsto \sum_{i<|w|}(1+w(i))2^{|w|-1-i}$, which is a variant of the dyadic encoding. Hence notations, relations, and operations for Σ^* are transferred to $\mathbb{N}$ and vice versa. In particular, $|n|$ denotes the length of $n \in \mathbb{N}$. Note $x \leq y$ for $x, y \in \mathbb{N}$ if and only if $u \leq v$ for the words u and v that correspond to x and y, respectively. Let $A \subseteq \Sigma^*$. Since we identify Σ^* and $\mathbb{N}$, we may consider the characteristic function of A as a function $\mathbb{N} \to \{0,1\}$, which is the characteristic sequence of A. Thus $A(x)$ denotes both the value of the characteristic function of A at point x and the x-th letter of the characteristic sequence of A, which are the same.

The identification of Σ^* and $\mathbb{N}$ gives rise to a few ambiguities, which, however, will always be eliminated by the context. We only give one general rule: always interpret the expressions 0^i and 1^i for $i \geq 2$ over Σ^*.

Oracle Turing Machines An oracle Turing machine is a Turing machine with an additional write-only tape, the so-called oracle tape, and additional states $q_?$, q_y, and q_n. The oracle tape has similar properties as the output tape: it is infinite to the right side and only to this side and in one computation step one symbol can be appended to the current word on the tape.

Let us explain the operating principles of oracle queries. The oracle Turing machine M is assigned some $O \subseteq \Sigma^*$. In this context we call O an *oracle*. Whenever the machine enters the state $q_?$, then until the beginning of the next step the following modifications and only these have been made:

- If the word currently on the oracle tape is in O, then M switches to the state q_y.
- If the word currently on the oracle tape is not in O, then M switches to the state q_n.
- The oracle tape is erased and the head points at the leftmost cell of the tape.

Thus roughly speaking, an oracle Turing machine may use some problem as a black box, ask arbitrary questions to it, and receives correct answers in one step. Let us emphasize that we use the unrestricted oracle model.[2]

[2]Two reasonable and widely used restrictions are:

- Only oracle queries of at most polynomial length are allowed.
- Nondeterministic branches are not allowed unless the oracle tape is empty.

We do without these restrictions, because there is no setting in this thesis in which the restrictions make a difference. In Chapter 3 only polynomial-time Turing machines have access to oracles. These clearly ask only queries of at most polynomial length. Forbidding them to branch while writing on the oracle tape makes no difference, because the machines —in case they are nondeterministic— can instead first nondeterministically guess the query and then write it on the oracle tape in a deterministic procedure. In Chapter 4 there is only

If M is an oracle Turing machine and $O \subseteq \Sigma^*$, then we denote by $M^O(x)$ the computation of M on input x with oracle O. All notions introduced for Turing machines without oracle access are analogously defined for oracle Turing machines. In particular, $M^O(x)$ accepts if and only if it has at least one accepting computation path and $L(M^O) = \{x \mid M^O(x) \text{ accepts}\}$. Again, we identify F^O for an oracle transducer F with the function the machine computes when given access to the oracle O and thus the notation $F^O(x)$ is ambiguous. Depending on the context, it refers to the computation of F on input x with access to the oracle O or it denotes this computation's output.

Enumerations of Turing Machines Note that there exists a universal Turing machine, i.e., a Turing machine that when given an encoding of some Turing machine M and some word x as input simulates M on input x and needs only polynomial time for each computation step of M. By this fact and by using clocked Turing machines, we obtain that there exist standard enumerations of Turing machines having the following properties.

Definition 2.3.1 *Let $A \subseteq \mathbb{N}^+$ be infinite. A sequence $(M_i)_{i \in A}$ is called a* standard enumeration *of nondeterministic, polynomial-time oracle Turing machines if it has the following properties:*

1. *All M_i are nondeterministic, polynomial-time oracle Turing machines.*
2. *For all oracles O and all inputs x the computation $M_i^O(x)$ stops within $|x|^i + i$ steps.*
3. *For every nondeterministic, polynomial-time oracle Turing machine M there exist infinitely many $i \in \mathbb{N}$ such that for all oracles O it holds $L(M^O) = L(M_i^O)$.*
4. *There exists a nondeterministic, polynomial-time oracle Turing machine M such that for all oracles O and all inputs x it holds that $M^O(\langle i, 0^{|x|^i+i}, x\rangle)$ nondeterministically simulates the computation $M_i^O(x)$.*

Analogously we define standard enumerations of polynomial-time oracle Turing transducers.

Throughout this thesis, we use the following standard enumerations:

- Let $M_1, M_2, \ldots$ be a standard enumeration of nondeterministic polynomial-time oracle Turing machines.
- Let $F_1, F_2, \ldots$ be a standard enumeration of polynomial time oracle Turing transducers.

2.3.2 Complexity Classes and Function Classes

Since in this thesis we get along with only quite a few complexity classes, we do not define generic complexity classes first, but instead directly define the relevant classes.

Recall that we identify Σ^* with $\mathbb{N}$. In the present section we define the classes as sets of languages, i.e., as subsets of Σ^*, or as classes of functions $\Sigma^* \to \Sigma^*$. Nevertheless, we may elsewhere consider them as subsets of $\mathbb{N}$ or functions $\mathbb{N} \to \mathbb{N}$. The existence of the pairing function defined in the next paragraph shows that our definitions also cover problems $\subseteq (\Sigma^*)^m$ (resp., $\mathbb{N}^m$) and functions $(\Sigma^*)^m \to (\Sigma^*)^n$ (resp., $\mathbb{N}^m \to \mathbb{N}^n$) for $m, n \in \mathbb{N}^+$.

one situation in which an oracle is used and in this case it is used by a deterministic logarithmic-space Turing machine, which can only ask queries of at most polynomial length (if it has a super-polynomial computation path, then this path is infinite and it may generate more than polynomially many symbols on the oracle tape, but it cannot ask the query) and clearly does bot branch at all.

Pairing Function and Encodings of Objects Let $\langle\cdot\rangle\colon \bigcup_{i\geq 0}\mathbb{N}^i \to \mathbb{N}$ be an injective, logarithmic-space computable, logarithmic-space invertible function such that $|\langle u_1,\dots,u_n\rangle| = 2(|u_1|+\cdots+|u_n|+n)$. Recall that $\langle\cdot\rangle$ can also be considered as a function $\bigcup_{i\geq 0}(\Sigma^*)^i \to \Sigma^*$ and that $\langle\cdot\rangle$ is polynomial-time computable and polynomial-time invertible as $\mathrm{FL}\subseteq\mathrm{FP}$ and $\mathrm{L}\subseteq\mathrm{P}$.

Most objects occurring in this thesis are of the following type $\mathcal{T}$, which we define inductively:

- All elements of $\mathbb{N}$ and Σ^* are objects of type $\mathcal{T}$.
- Finite tuples and finite sets of objects of type $\mathcal{T}$ are also objects of type $\mathcal{T}$.

Note that $\mathcal{T}$ is an ad hoc notation designed only for this paragraph.

As finite sets can always be encoded as tuples (i.e., finite lists), the pairing function defines an encoding and in particular the length of all objects of type $\mathcal{T}$. Note that thus not each object has a unique encoding (e.g., for a finite set of cardinality $n\in\mathbb{N}$ there are 2^n ways to encode the set as a list), but all encodings of a certain object have the same length, which we will denote by $|\cdot|$.

Without loss of generality, we may assume that all graphs are objects of type $\mathcal{T}$. Moreover, all functions mapping from a finite set of objects of type $\mathcal{T}$ to another finite set of objects of type $\mathcal{T}$ are of type $\mathcal{T}$ themselves. Hence in particular, encodings and lengths of integer circuits and all other relevant objects occurring in Section 4 are now defined.

Note that we do not define appropriate encodings for all kinds of objects. Due to that and often just for the sake of simplicity, we may also consider sets of tuples, graphs, or other objects as members of the below classes.

Logarithmic-Space Classes Define

$$\begin{aligned}
\mathrm{FL} &= \{f\colon \Sigma^*\to\Sigma^* \mid f \text{ is total and logarithmic-space computable}\}\\
\mathrm{L} &= \{L\subseteq\Sigma^* \mid \text{there is a deterministic logarithmic-space Turing machine that decides } L\}\\
\mathrm{NL} &= \{L\subseteq\Sigma^* \mid \text{there is a nondeterministic logarithmic-space Turing machine accepting } L\}
\end{aligned}$$

Note that it is commonly known that FL is closed under function composition, i.e., for all $f,f'\in\mathrm{FL}$ the function $f\circ f'$ defined by $x\mapsto f(f'(x))$ is in FL as well.

Polynomial-Time Classes We define

$$\begin{aligned}
\mathrm{FP} &= \{f\colon \Sigma^*\to\Sigma^* \mid f \text{ is total and polynomial-time computable}\}\\
\mathrm{P} &= \{L\subseteq\Sigma^* \mid \text{there is a deterministic polynomial-time Turing machine that decides } L\}\\
\mathrm{UP} &= \{L\subseteq\Sigma^* \mid \text{there is a nondeterministic polynomial-time Turing machine } M \text{ accepting } L\\
&\qquad\qquad \text{such that for each } x\in\Sigma^*,\ M(x) \text{ has at most one accepting path}\}\\
\mathrm{NP} &= \{L\subseteq\Sigma^* \mid \text{there is a nondeterministic polynomial-time Turing machine accepting } L\}
\end{aligned}$$

Further Classes Let us define

$$\begin{aligned}
\mathrm{FPSPACE} &= \{f\colon \Sigma^*\to\Sigma^* \mid f \text{ is total and polynomial-space computable}\}\\
\mathrm{PSPACE} &= \{L\subseteq\Sigma^* \mid \text{there exists a deterministic polynomial-space Turing machine that}\\
&\qquad\qquad \text{decides } L\}\\
\mathrm{REC} &= \{L\subseteq\Sigma^* \mid L \text{ is decidable}\}\\
\mathrm{RE} &= \{L\subseteq\Sigma^* \mid L \text{ is computably enumerable}\}
\end{aligned}$$

There are known a couple of characterizations for RE. We only mention two: (i) RE is the set of those languages whose semi-characteristic function is computable and (ii) RE is the set of those languages L for which there exists $L' \in \mathrm{REC}$ with $L = \{x \mid \exists_{y \in \Sigma^*} \langle x, y\rangle \in L'\}$ (i.e., L is the *projection* of a decidable set).

It holds $\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{REC} \subseteq \mathrm{RE}$ and $\mathrm{FL} \subseteq \mathrm{FP} \subseteq \mathrm{FPSPACE}$. For proofs of the non-trivial inclusions we refer to textbooks such as [Pap94, AB09].

Complement Classes For an arbitrary complexity class $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ we define $\mathrm{co}\mathcal{C} = \{L \subseteq \Sigma^* \mid \overline{L} \in \mathcal{C}\}$. All the deterministic time and space classes above are closed under complement, i.e., for such a class $\mathcal{C}$ it holds $\mathcal{C} = \mathrm{co}\mathcal{C}$. For nondeterministic time and space classes this is not known in general and for many classes even widely believed not to be true. Nevertheless, a result due to Immerman [Imm88] and Szelepcsényi [Sze88] yields that a wide range of nondeterministic space classes is indeed closed under complement. One of these classes is NL, i.e., it holds $\mathrm{NL} = \mathrm{coNL}$.

Relativized Complexity Classes Now let us define some of the classes above relative to an oracle $O \subseteq \Sigma^*$. We start again with logarithmic-space classes, which are —apart from this chapter— only relevant to Chapter 4.

$$\begin{aligned}
\mathrm{FL}^O &= \{f\colon \Sigma^* \to \Sigma^* \mid f \text{ total, } f = F^O \text{ for a logarithmic-space oracle Turing transducer F}\}\\
\mathrm{L}^O &= \{L \subseteq \Sigma^* \mid \text{there exists a deterministic logarithmic-space oracle Turing machine } M \\
&\qquad \text{with } L(M^O) = L\}
\end{aligned}$$

Next we define relativized polynomial-time classes.

$$\begin{aligned}
\mathrm{FP}^O &= \{f\colon \Sigma^* \to \Sigma^* \mid f \text{ total, } f = F^O \text{ for a polynomial-time oracle Turing transducer F}\}\\
\mathrm{P}^O &= \{L \subseteq \Sigma^* \mid \text{there is a deterministic polynomial-time oracle Turing machine } M \text{ with} \\
&\qquad L(M^O) = L\}\\
\mathrm{UP}^O &= \{L \subseteq \Sigma^* \mid \text{there is a nondeterministic polynomial-time oracle Turing machine } M \text{ such} \\
&\qquad \text{that (i) } L(M^O) = L \text{ and (ii) } \forall_{x \in \Sigma^*} M^O(x) \text{ has at most one accepting path}\}\\
\mathrm{NP}^O &= \{L \subseteq \Sigma^* \mid \text{there exists a nondeterministic polynomial-time oracle Turing machine } M \\
&\qquad \text{with } L(M^O) = L\}
\end{aligned}$$

The following inclusions hold relative to all oracles O.

$$\mathrm{L}^O \subseteq \mathrm{P}^O \subseteq \mathrm{UP}^O \subseteq \mathrm{NP}^O \quad \text{and} \quad \mathrm{FL}^O \subseteq \mathrm{FP}^O.$$

For proofs of the non-trivial of these inclusions we refer to the proofs for the corresponding unrelativized classes mentioned above. These can all be relativized.

For classes $\mathcal{C}$ and $\mathcal{C}'$ we define $(\mathrm{co}\mathcal{C})^O = \mathrm{co}(\mathcal{C}^O)$ and $(\mathcal{C} \cap \mathcal{C}')^O = \mathcal{C}^O \cap \mathcal{C}'^O$. Moreover, let $\mathcal{C}^{\mathcal{C}'} = \bigcup_{O \in \mathcal{C}'} \mathcal{C}^O$. The following proposition basically follows from $\mathrm{NL} = \mathrm{coNL}$.

Proposition 2.3.2 $\mathrm{L}^{\mathrm{NL}} = \mathrm{NL}$.

Finally, note that for every oracle O, the sequence $(M_i)_{i \in \mathbb{N}^+}$ represents an enumeration of the languages in NP^O, i.e., $\mathrm{NP}^O = \{L(M_i^O) \mid i \in \mathbb{N}^+\}$. Analogously, $\mathrm{FP}^O = \{F_i^O \mid i \in \mathbb{N}^+\}$.

2.3.3 Reducibilities and Complete Problems

Reducibilities We define several reducibilities. Let $A, B, O \subseteq \Sigma^*$.

- A is *polynomially many-one reducible* to B, denoted by $A \leq_{\mathrm{m}}^{\mathrm{p}} B$, if there exists $f \in \mathrm{FP}$ such that for all $x \in \Sigma^*$ it holds $x \in A \Leftrightarrow f(x) \in B$.

- A is *polynomially many-one reducible to B relative to O*, denoted by $A \leq_{\mathrm{m}}^{\mathrm{p},O} B$, if there exists $f \in \mathrm{FP}^O$ such that for all $x \in \Sigma^*$ it holds $x \in A \Leftrightarrow f(x) \in B$.

- A is *logarithmic-space many-one reducible* to B, denoted by $A \leq_{\mathrm{m}}^{\log} B$, if there exists $f \in \mathrm{FL}$ such that for all $x \in \Sigma^*$ it holds $x \in A \Leftrightarrow f(x) \in B$.

- A is *many-one reducible* to B, denoted by $A \leq_{\mathrm{m}} B$, if there exists a total and computable $f\colon \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ it holds $x \in A \Leftrightarrow f(x) \in B$.

For $A, B \subseteq \Sigma^*$ and $\prec$ denoting one of the reducibilities above we say that A and B are $\prec$-equivalent if $A \prec B$ and $B \prec A$.

Note that in Section 2.3.5 we define separate reducibilities for disjoint pairs. These can be seen as generalizations of corresponding reducibilities for problems $\subseteq \Sigma^*$. Moreover, Section 2.3.6 contains the definition of a further reducibility, namely one for total polynomial search problems.

Hardness and Completeness Let $\prec$ denote one of the reducibilities above and $\mathcal{C}$ be some complexity class. A is called *$\prec$-hard* for $\mathcal{C}$ if $C \prec A$ for all $C \in \mathcal{C}$. Moreover, A is called *$\prec$-complete* for $\mathcal{C}$ if $A \in \mathcal{C}$ and A is $\prec$-hard for $\mathcal{C}$.

Some Prominent Problems We first define some variations of the standard graph accessibility problem. For $k \in \mathbb{N}^+$ define

$$\begin{aligned} \mathrm{GAP}_{\geq k} &= \{(V, E, s, t) \mid (V, E) \text{ is a directed multigraph, } s, t \in V, \text{ there exist at least } k \\ &\qquad\qquad \text{paths from } s \text{ to } t \text{ in } (V, E)\} \\ \mathrm{GAP}_{=k} &= \mathrm{GAP}_{\geq k} \cap \overline{\mathrm{GAP}_{\geq k+1}}. \end{aligned}$$

$\mathrm{GAP}_{\geq 1}$ is the standard graph accessibility problem for directed multigraphs. It is $\leq_{\mathrm{m}}^{\log}$-complete for NL. For a proof we refer to [AB09].[3] In this thesis, we only need the membership of $\mathrm{GAP}_{\geq 1}$ in NL. A similar argument as in the mentioned proof in [AB09] shows that also $\mathrm{GAP}_{\geq k}$ for arbitrary $k \in \mathbb{N}^+$ is in NL. By the closure properties of NL, it holds $\mathrm{GAP}_{=k} \in \mathrm{NL}$ for all $k \in \mathbb{N}^+$ as well.

Let SAT (resp., TAUT) be the set of all satisfiable (resp., tautological) propositional formulas (encoded in some standard way). SAT (resp., TAUT) is known to be $\leq_{\mathrm{m}}^{\log}$-complete for NP (resp., coNP).[4] The circuit version of SAT is also $\leq_{\mathrm{m}}^{\log}$-complete for NP and is precisely defined in the beginning of Section 4.3.2.

Canonical Complete Problems Let $O \subseteq \Sigma^*$ and define

$$\mathcal{CAN}^O = \{\langle 0^i, 0^t, x\rangle \mid i, t, x \in \mathbb{N}, i > 0, \text{ and } M_i^O(x) \text{ accepts within } t \text{ steps}\}.$$

By the properties of standard enumerations, the problem is in NP^O. A simple reduction shows that $\mathcal{CAN}^O$ is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete for NP^O. Then clearly $\overline{\mathcal{CAN}^O}$ is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete for coNP^O.

[3] Indeed, only the graph accessibility problem for directed graphs is proven to be NL-complete there. However, the proof also shows that $\mathrm{GAP}_{\geq 1}$, the accessibility problem for directed multigraphs, is NL-complete.

[4] For a proof we refer to [Coo71]. Indeed, a simple adaption of Cook's proof is needed to obtain the $\leq_{\mathrm{m}}^{\log}$-completeness of SAT for NP.

2.3.4 Proof Systems

Definition 2.3.3 ([CR79]) *A function $f \in \mathrm{FP}$ is called a* proof system *for the set* $\mathrm{ran}(f)$*. If $f(x) = y$, then we say that x is an f-proof for y. Moreover, $f \in \mathrm{FP}$ is a* propositional proof system (pps) *if* $\mathrm{ran}(f) = \mathrm{TAUT}$.

Let f and g be total functions. We say that f is simulated by *g, denoted by $f \leq g$, if there exists a total function π and a polynomial p such that $|\pi(x)| \leq p(|x|)$ and $g(\pi(x)) = f(x)$ for all x. Moreover, f* is P-simulated by *g, denoted by $f \leq^{\mathrm{p}} g$, if there exists a function $\pi \in \mathrm{FP}$ such that $g(\pi(x)) = f(x)$ for all x.*

A function $g \in \mathrm{FP}$ is called optimal *or* length-optimal *if $f \leq g$ for all $f \in \mathrm{FP}$ with* $\mathrm{ran(f)} = \mathrm{ran(g)}$*. A function $g \in \mathrm{FP}$ is* P-optimal *if $f \leq^{\mathrm{p}} g$ for all $f \in \mathrm{FP}$ with* $\mathrm{ran(f)} = \mathrm{ran(g)}$*.*

We define the corresponding relativized notions. For that purpose let $O \subseteq \Sigma^*$.

Definition 2.3.4 *A function $f \in \mathrm{FP}^O$ is called a* proof system *for the set* $\mathrm{ran}(f)$ relative to O*.*

For $f, g \in \mathrm{FP}^O$ we say that f is P^O-simulated by *g, denoted by $f \leq^{\mathrm{p},O} g$, if there exists a function $\pi \in \mathrm{FP}^O$ such that $g(\pi(x)) = f(x)$ for all x.*

A function $g \in \mathrm{FP}^O$ is called optimal relative to O *or* length-optimal relative to O *if $f \leq g$ for all $f \in \mathrm{FP}^O$ with $\mathrm{ran}(f) = \mathrm{ran}(g)$. A function $g \in \mathrm{FP}^O$ is* P^O-optimal *if $f \leq^{\mathrm{p},O} g$ for all $f \in \mathrm{FP}^O$ with $\mathrm{ran}(f) = \mathrm{ran}(g)$.*

Assuming no confusions will arise, a function $f \in \mathrm{FP}^O$ will often just be called a proof system for $\mathrm{ran}(f)$ instead of a proof system for $\mathrm{ran}(f)$ *relative to* O. Analogously, instead of speaking of optimal proof systems *relative to* O, we will often simply speak of optimal proof systems.

The following proposition states the relativized version of a relativizably proven result by Köbler, Messner, and Torán [KMT03].

Proposition 2.3.5 ([KMT03]) *1. For every oracle O, if A has an optimal proof system relative to O and $\emptyset \neq B \leq_{\mathrm{m}}^{\mathrm{p},O} A$, then B has an optimal proof system relative to O.*

2. For every oracle O, if A has a P^O-optimal proof system and $\emptyset \neq B \leq_{\mathrm{m}}^{\mathrm{p},O} A$, then B has a P^O-optimal proof system.

Corollary 2.3.6 *1. For every oracle O, if there exists a $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete $A \in \mathrm{NP}^O$ (resp., $A \in \mathrm{coNP}^O$) that has optimal proof systems relative to O, then all non-empty sets in NP^O (resp., coNP^O) have optimal proof systems relative to O.*

2. For every oracle O, if there exists a $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete $A \in \mathrm{NP}^O$ (resp., $A \in \mathrm{coNP}^O$) that has P^O-optimal proof systems, then all non-empty sets in NP^O (resp., coNP^O) have P^O-optimal proof systems.

Remark 2.3.7 ([DG19]) *The notion of propositional proof systems does not have a canonical relativization. However, in the light of Corollary 2.3.6, it is reasonable to use the subsequent convention. Let $O \subseteq \Sigma^*$.*

- *There exist optimal propositional proof systems relative to O if and only if there is some A which is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete for coNP^O and has optimal proof systems relative to O.*
- *There exist P^O-optimal propositional proof systems if and only if there is some A which is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete for coNP^O and has P^O-optimal proof systems.*

2.3.5 Disjoint Pairs

Unless stated differently, we only define relativized notions in this subsection. However, the unrelativized notions are obtained if the oracle O is chosen to be say the empty set. In such cases, for all notations we simply omit the oracle. Let $O \subseteq \Sigma^*$.

Definition 2.3.8 *If $A, B \in \mathrm{NP}^O$ (resp., $A, B \in \mathrm{coNP}^O$) with $A \cap B = \emptyset$, then we call (A, B) a* disjoint NP^O*-pair (resp.,* disjoint coNP^O*-pair). The set of all disjoint* NP*-pairs (resp.,* coNP*-pairs) is denoted by* DisjNP^O *(resp.,* $\mathrm{DisjCoNP}^O$*).*

There are several reducibilities for disjoint pairs. In this thesis we only make use of two different reducibilities: polynomial many-one reducibility and the many-one reducibility allowing unbounded complexity. We start with the standard polynomial-time many-one reducibility.

Definition 2.3.9 ([Raz94]) *Let $A, B, C, D \subseteq \Sigma^*$ such that $A \cap B = C \cap D = \emptyset$. (C, D) is* polynomially many-one reducible *to (A, B) relative to O, denoted by $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$, if there exists an $f \in \mathrm{FP}^O$ with $f(C) \subseteq A$ and $f(D) \subseteq B$. If $C = \overline{D}$, then we also write $C \leq_{\mathrm{m}}^{\mathrm{p},O} (A, B)$ instead of $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$. If $A = \overline{B}$, then we also write $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} A$ instead of $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$.*

We prefer the notation $C \leq_{\mathrm{m}}^{\mathrm{p},O} (A, B)$ to $C \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$ as in this case there is no promise involved.

Definition 2.3.10 *We say that a disjoint pair (A, B) is $\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-hard *($\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-complete*) for DisjNP^O if $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$ for all $(C, D) \in \mathrm{DisjNP}^O$ (and $(A, B) \in \mathrm{DisjNP}^O$).*

Analogously, a disjoint pair (A, B) is $\leq_{\mathrm{m}}^{\mathrm{pp},O}$-hard *($\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-complete*) for $\mathrm{DisjCoNP}^O$ if $(C, D) \leq_{\mathrm{m}}^{\mathrm{pp},O} (A, B)$ for all $(C, D) \in \mathrm{DisjCoNP}^O$ (and $(A, B) \in \mathrm{DisjCoNP}^O$).*

Moreover, a pair (A, B) is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-hard *for some complexity class $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ if $C \leq_{\mathrm{m}}^{\mathrm{p},O} (A, B)$ for every $C \in \mathcal{C}$.*

The second reducibility we consider is the standard many-one reducibility. Here we only need the unrelativized notion.

Definition 2.3.11 *Let $A, B, C, D \subseteq \Sigma^*$ such that $A \cap B = C \cap D = \emptyset$. (C, D) is* many-one reducible *to (A, B), denoted by $(C, D) \leq_{\mathrm{m}} (A, B)$, if there exists a computable, total function f with $f(C) \subseteq A$ and $f(D) \subseteq B$. If $C = \overline{D}$, then we also write $C \leq_{\mathrm{m}} (A, B)$ instead of $(C, D) \leq_{\mathrm{m}} (A, B)$. If $A = \overline{B}$, then we also write $(C, D) \leq_{\mathrm{m}} A$ instead of $(C, D) \leq_{\mathrm{m}} (A, B)$.*

As this reducibility is a promise reducibility, it would be more consequent to write $\leq_{\mathrm{m}}^{\mathrm{p}}$ instead of $\leq_{\mathrm{m}}$. However, we do not do that in order to avoid confusions.

The following theorem establishes a connection between the notion of proof systems and the notion of disjoint NP-pairs.

Theorem 2.3.12 ([Raz94][5]) *Let $O \subseteq \Sigma^*$. If there exist optimal pps relative to O, then DisjNP^O has $\leq_{\mathrm{m}}^{\mathrm{pp},O}$-complete elements.*

For a straightforward and relativizable proof of the unrelativized version of this theorem we refer to [GSSZ04].

[5] Note that there are contradicting statements on the origin of the unrelativized result in the literature (cf. [KMT03, GSSZ04, Pud17] for example).

2.3.6 Total Polynomial Search Problems

Total polynomial search problems (TFNP problems for short) are more commonly known under the name total NP search problems. Nevertheless, we prefer the former name.

Again we only define the relativized notion and point out that the unrelativized notion is obtained by choosing the oracle O to be the empty set for instance. Let $O \subseteq \Sigma^*$.

Definition 2.3.13 ([MP91]) *A* total polynomial search problem relative to O *is a pair* (p, R) *where* p *is a polynomial and* $R \in \mathrm{P}^O$ *with (i)* $\forall_x \exists_y \left(|y| \leq p(|x|) \wedge \langle x, y\rangle \in R\right)$ *and (ii)* $\forall_x \forall_y \left(|y| > p(|x|) \Rightarrow \langle x, y\rangle \notin R\right)$. *The computational task is: on input* x *and having access to the oracle* O, *compute some* y *with* $|y| \leq p(|x|) \wedge \langle x, y\rangle \in R$. *Let* TFNP^O *be the set of all total polynomial search problems relative to* O.

If the oracle O is apparent from the context, we may speak of TFNP problems instead of TFNP problems *relative to* O. Let us sketch two equivalent ways of defining TFNP problems.

First, as mentioned in the introduction, a total polynomial search problem can be also seen as the following computational task: for a fixed nondeterministic multivalued function with values of polynomial length that are polynomially verifiable and guaranteed to exist [MP91], determine on input x some element of the set of all values that the function has at point x.

Second, a total polynomial search problem can be considered as the computational task to determine on input x an accepting path of the computation $M(x)$ where M is a fixed nondeterministic polynomial-time Turing machine M that accepts Σ^*.

Let us define the standard polynomial-time many-one reducibility for TFNP problems.

Definition 2.3.14 ([JPY88]) *A TFNP problem* (p, R) *is polynomially many-one reducible to a TFNP problem* (q, S) *relative to* O *if there are* $f, g \in \mathrm{FP}^O$ *such that*

$$\forall_x \forall_z \left(S(\langle f(x), z\rangle) \Rightarrow R(x, g(\langle x, z\rangle))\right).$$

A TFNP problem is polynomially many-one complete relative to O *if all problems in* TFNP^O *are polynomially many-one reducible to it relative to* O. *As an abbreviation, we say that* TFNP^O *has (many-one) complete problems if and only if* TFNP^O *contains an element that is polynomially many-one complete relative to* O.

Roughly speaking, (p, R) is polynomially many-one reducible to (q, S) relative to O if and only if (p, R) can be solved by a deterministic polynomial-time algorithm with access to the oracle O that may also ask one oracle query to the problem (q, S) and is given some correct answer in one step. If we allow not only one but arbitrarily many queries to the problem (q, S), we obtain a stronger reducibility notion: the polynomial reducibility (cf. e.g. [Pud17]).

As relative to all oracles there are complete TFNP problems with respect to the polynomial reducibility if and only if there are many-one complete TFNP problems [Pud17, Proposition 4.10][6], we go without defining other reducibilities formally.

Let us finally formulate a connection between complete TFNP problems and both P-optimal proof systems and complete disjoint coNP-pairs.

Theorem 2.3.15 ([BKM09, Pud17]) *Let* $O \subseteq \Sigma^*$. *The following statements hold.*

1. *If there exists an* NP^O*-complete problem that has* P^O*-optimal proof systems, then* TFNP^O *contains polynomially many-one complete problems relative to* O.
2. *If* TFNP^O *contains problems that are polynomially many-one complete relative to* O, *then* DisjCoNP *contains* $\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-complete pairs.*

[6] Pudlák [Pud17] mentions that this result is due to Emil Jeřábek and presents a relativizable proof of it.

Chapter 3

Separating Relativized Conjectures

In this chapter we construct oracles. As was mentioned in the introduction of this thesis, these oracles address several major conjectures listed by Pudlák [Pud17] and make progress in a working program initiated by him.

In Section 3.1 we formulate these conjectures formally and introduce some notions and notations useful for constructing oracles. Then in each of the Sections 3.2, 3.3, 3.4, and 3.5 we build one oracle. Finally, in Section 3.6 we present the current state of the art regarding known oracle separations between the conjectures listed by Pudlák and list questions for further research.

3.1 Basic Definitions and Outline

In this section we first define the conjectures that are this chapter's central object of study. Then some technical notations and results are presented that find use in most of the sections in this chapter.

3.1.1 Conjectures

Let us define the main conjectures this chapter deals with. These have already been explained in the introduction, but now all relevant notions have been formally defined, which enables us to define the conjectures formally. We first define the unrelativized conjectures and then list the corresponding relativized statements. Assuming the context will resolve any ambiguities, the names of the unrelativized statements will be used for the corresponding relativized conjectures as well.

$$
\begin{aligned}
\mathsf{CON} &:= \text{P-optimal propositional proof systems do not exist} \\
\mathsf{SAT} &:= \text{SAT does not have P-optimal proof systems} \\
\mathsf{CON}^{\mathsf{N}} &:= \text{Optimal propositional proof systems do not exist} \\
\mathsf{TFNP} &:= \text{TFNP does not contain polynomially many-one complete problems} \\
\mathsf{DisjNP} &:= \text{DisjNP does not contain } \leq_{\mathrm{m}}^{\mathrm{pp}}\text{-complete pairs} \\
\mathsf{DisjCoNP} &:= \text{DisjCoNP does not contain } \leq_{\mathrm{m}}^{\mathrm{pp}}\text{-complete pairs} \\
\mathsf{NP} \cap \mathsf{coNP} &:= \text{NP} \cap \text{coNP does not contain } \leq_{\mathrm{m}}^{\mathrm{p}}\text{-complete problems} \\
\mathsf{UP} &:= \text{UP does not contain } \leq_{\mathrm{m}}^{\mathrm{p}}\text{-complete problems} \\
\mathsf{CON} \vee \mathsf{SAT} &:= \mathsf{CON} \text{ or } \mathsf{SAT} \text{ holds}
\end{aligned}
$$

A further important conjecture, indeed the most famous conjecture in computational complexity theory at all, is P $\neq$ NP, which is also relevant to this chapter, but for which we do not introduce any other notation. Recall that by Corollary 2.3.6, it holds

$$\begin{aligned}
\mathsf{CON} &\iff \text{coNP does not contain a } \leq^{\mathrm{p}}_{\mathrm{m}}\text{-complete set that has P-optimal proof systems}\\
\mathsf{CON}^{\mathsf{N}} &\iff \text{coNP does not contain a } \leq^{\mathrm{p}}_{\mathrm{m}}\text{-complete set that has optimal proof systems}\\
\mathsf{SAT} &\iff \text{NP does not contain a } \leq^{\mathrm{p}}_{\mathrm{m}}\text{-complete set that has P-optimal proof systems}
\end{aligned}$$

Let us now list the relativized statements that correspond to the above conjectures. For the sake of simplicity, we will call them relativized *conjectures* although this might be misunderstandable as the statements have only been conjectured in the unrelativized case (i.e., relative to —for instance— the empty oracle) and for each of these statements there exist oracles relative to which it does not hold.

Let O be some oracle. We formulate the conjectures relative to O. For most of the conjectures it is straightforward to relativize them. For the conjectures associated with proof systems we refer to Corollary 2.3.6 and Remark 2.3.7 for the reasons that lead to the statements below.

$$\begin{aligned}
\mathsf{CON} &:= \mathrm{P}^O\text{-optimal propositional proof systems do not exist}\\
\mathsf{SAT} &:= \mathrm{NP}^O \text{ does not contain a } \leq^{\mathrm{p},O}_{\mathrm{m}}\text{-complete set having P}^O\text{-optimal proof systems}\\
\mathsf{CON}^{\mathsf{N}} &:= \text{Optimal propositional proof systems relative to } O \text{ do not exist}\\
\mathsf{TFNP} &:= \mathrm{TFNP}^O \text{ does not contain polynomially many-one complete problems}\\
\mathsf{DisjNP} &:= \mathrm{DisjNP}^O \text{ does not contain } \leq^{\mathrm{pp},O}_{\mathrm{m}}\text{-complete pairs}\\
\mathsf{DisjCoNP} &:= \mathrm{DisjCoNP}^O \text{ does not contain } \leq^{\mathrm{pp},O}_{\mathrm{m}}\text{-complete pairs}\\
\mathsf{NP} \cap \mathsf{coNP} &:= \mathrm{NP}^O \cap \mathrm{coNP}^O \text{ does not contain } \leq^{\mathrm{p},O}_{\mathrm{m}}\text{-complete problems}\\
\mathsf{UP} &:= \mathrm{UP}^O \text{ does not contain } \leq^{\mathrm{p},O}_{\mathrm{m}}\text{-complete problems}\\
\mathsf{CON} \vee \mathsf{SAT} &:= \mathsf{CON} \text{ or } \mathsf{SAT} \text{ holds (relative to } O)
\end{aligned}$$

Furthermore, NP $\neq$ P is relativized as $\mathrm{NP}^O \neq \mathrm{P}^O$. Let us again formulate some characterizations.

$$\begin{aligned}
\mathsf{CON} &\iff \mathrm{coNP}^O \text{ does not contain a } \leq^{\mathrm{p},O}_{\mathrm{m}}\text{-complete set having P}^O\text{-optimal proof systems}\\
\mathsf{CON}^{\mathsf{N}} &\iff \text{coNP does not contain a } \leq^{\mathrm{p},O}_{\mathrm{m}}\text{-complete set having optimal proof systems}
\end{aligned}$$

3.1.2 Some Notions Designed for Building Oracles

Let us introduce some notions that have proven beneficial for construction oracles. These notions originate from [DG19].

An oracle $O \subseteq \mathbb{N}$ is identified with its characteristic sequence $O(0)O(1)\cdots$. A finite word w describes an oracle that is partially defined and thus also called a *partial oracle*, i.e., it is only defined for natural numbers $x < |w|$. Occasionally, we use w instead of the set $\{i \mid w(i) = 1\}$ and write for example $A = w \cup B$ where A and B are sets. In particular, for nondeterministic oracle Turing machines M and deterministic oracle Turing transducers F, the notations $M^w(x)$ and $F^w(x)$ refer to $M^{\{i|w(i)=1\}}(x)$ and $F^{\{i|w(i)=1\}}(x)$ (hence oracle queries that w is not defined for are answered by "no"). Using w instead of $\{i \mid w(i) = 1\}$ additionally allows us to define the following notion: the computation $M^w(x)$ *definitely accepts* if it contains an accepting path and all queries on this path are $< |w|$. The computation $M^w(x)$ *definitely rejects* if all paths reject and all queries are $< |w|$. We say that the computation $M^w(x)$ *is definite* if it definitely accepts or definitely rejects. Similarly, the computation $F^w(x)$ is *definite* if all its queries are $< |w|$.

The following lemma is used multiple times in this chapter, which is the reason why it occurs in this general section. Roughly speaking, it shows that if a partial oracle is defined for some word, then extending the oracle does not change this word's membership in the canonical NP-complete problem. Of course, the analogous holds for the canonical coNP-complete problem, which is the complement of the canonical NP-complete problem.

Lemma 3.1.1 ([DG19]) *For all $y \leq |w|$ and all $v \sqsupseteq w$ it holds $(y \in \mathcal{CAN}^v \Leftrightarrow y \in \mathcal{CAN}^w)$.*

Proof We may assume $y = \langle 0^i, 0^t, x\rangle$ for suitable $i \in \mathbb{N}^+$ and $t, x \in \mathbb{N}$, since otherwise, $y \notin \mathcal{CAN}^w$ and $y \notin \mathcal{CAN}^v$. For each q that is queried within the first t steps of $M_i^w(x)$ or $M_i^v(x)$ it holds that $|q| \leq t < |y|$ and thus $q < y$. Hence these queries are answered the same way relative to w and v, showing that $M_i^w(x)$ accepts within t steps if and only if $M_i^v(x)$ accepts within t steps. □

3.2 DisjNP, NP ∩ coNP, and ¬UP Relative to an Oracle

In this section we construct an oracle O relative to which (i) P = UP and hence UP has $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets, (ii) there exist no $\leq_{\mathrm{m}}^{\mathrm{pp}}$-complete disjoint NP-pairs, and (iii) NP ∩ coNP does not have $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets. The properties (ii) and (iii) can be subsumed under the stronger statement that relative to the oracle O, DisjNP does not contain pairs that are $\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for NP ∩ coNP.

This oracle makes some progress in the aforementioned working program initiated by Pudlák [Pud17]. Its main achievement is that it illustrates that for none of the implications DisjNP ⇒ UP, $\mathsf{CON}^{\mathsf{N}}$ ⇒ UP, and CON ⇒ UP there exists a relativizable proof. Note that the converse of the last of the three implications is true relative to all oracles[1]. Thus in some sense, the conjecture UP is strictly stronger than the conjecture CON: there does not exist an oracle relative to which UP ∧ ¬CON, but there does exist an oracle relative to which we have CON ∧ ¬UP.

Additionally, the oracle shows that the implication NP ∩ coNP ⇒ UP cannot be proven using solely relativizable techniques. This result will be extended in Section 3.4, where we show that even the weaker implication NP ∩ coNP ⇒ CON requires —in case it holds— non-relativizable proof techniques.

The following lemma yields a common tool that was already used by Hartmanis and Hemaspaandra [HH88]. We apply it multiple times in [DG19] and once within this thesis.

Lemma 3.2.1 *For all $i, j \in \mathbb{N}^+$ there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and all $D \subseteq \Sigma^*$ there exist an even $x \in \Sigma^n$ and an odd $y \in \Sigma^n$ such that at least one of the following statements holds.*

1. $M_i^{D\cup\{x\}}(0^n)$ *rejects*
2. $M_j^{D\cup\{y\}}(0^n)$ *rejects*
3. $M_i^{D\cup\{x,y\}}(0^n)$ *and* $M_j^{D\cup\{x,y\}}(0^n)$ *accept*

Proof Assume that the assertion is wrong, i.e., there are $i, j \in \mathbb{N}^+$ such that for all $n_0 \in \mathbb{N}$ there is an $n \geq n_0$ and an oracle $D \subseteq \Sigma^*$ such that for all even $x \in \Sigma^n$ and all odd $y \in \Sigma^n$ all three statements are wrong. Fix machines M_i and M_j guaranteed by this assumption.

[1] A relativizable proof is given in [KMT03].

Let p be the polynomial $\alpha \mapsto \alpha^{i+j} + i + j$. Note that p limits the running time of M_i and M_j. Choose n_0 such that $2^{2n_0-2} > 2^{n_0} \cdot p(n_0)$. Let $n \geq n_0$ and $D \subseteq \Sigma^*$ such that for all even $x \in \Sigma^n$ and all odd $y \in \Sigma^n$ the three statements are wrong.

As the first statement is wrong, for all even $x \in \Sigma^n$ the computation $M_i^{D\cup\{x\}}(0^n)$ accepts. Since the second statement is wrong as well, for all odd $y \in \Sigma^n$ the computation $M_j^{D\cup\{y\}}(0^n)$ accepts. Consider the set

$$\begin{aligned} E =& \{(x,y) \in (\Sigma^n)^2 \mid x \text{ even, } y \text{ odd, the least accepting path of } M_i^{D\cup\{x\}}(0^n) \text{ queries } y\} \cup \\ & \{(x,y) \in (\Sigma^n)^2 \mid x \text{ even, } y \text{ odd, the least accepting path of } M_j^{D\cup\{y\}}(0^n) \text{ queries } x\} \end{aligned}$$

Since by the choice of p, the number of oracle queries of each of the machines M_i and M_j is bounded by p, it holds $|E| \leq 2^n \cdot p(n) < 2^{2n-2} = |\{x \in \Sigma^n \mid x \text{ even}\} \times \{y \in \Sigma^n \mid y \text{ odd}\}|$. Hence there are $x \in \Sigma^n$ even and $y \in \Sigma^n$ odd such that $(x,y) \notin E$, i.e., the least accepting path of $M_i^{D\cup\{x\}}(0^n)$ does not query y and the least accepting path of $M_j^{D\cup\{y\}}(0^n)$ does not query x. Thus $M_i^{D\cup\{x,y\}}(0^n)$ and $M_j^{D\cup\{x,y\}}(0^n)$ both accept, which contradicts the assumption that statement 3 is wrong. □

Corollary 3.2.2 *For all $i, j, r \in \mathbb{N}^+$ there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and all $D \subseteq \Sigma^*$ there exist $x \in \Sigma^n$ even and $y \in \Sigma^n$ odd such that at least one of the following statements holds.*

1. $F_r^{D\cup\{x\}}(0^n) \notin L(M_i^{D\cup\{x\}})$
2. $F_r^{D\cup\{y\}}(0^n) \notin L(M_j^{D\cup\{y\}})$
3. $F_r^{D\cup\{x,y\}}(0^n) \in L(M_i^{D\cup\{x,y\}}) \cap L(M_j^{D\cup\{x,y\}})$

Proof The statement follows by applying Lemma 3.2.1 to the machines N_i and N_j that work as follows: N_i (resp., N_j) first computes $F_r(0^n)$ and then simulates M_i (resp., M_j) on input $F_r(0^n)$. □

Now we formulate and prove this section's main result. Let us remark that the proof adopts ideas by Baker, Gill, and Solovay [BGS75] and Rackoff [Rac82]. In particular, the way we achieve $\mathrm{P}^O = \mathrm{UP}^O$ is based on ideas in these articles.

Theorem 3.2.3 *There exists an oracle O with the following properties.*

1. DisjNP^O *does not have pairs that are* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-hard for* $\mathrm{NP}^O \cap \mathrm{coNP}^O$.
2. $\mathrm{P}^O = \mathrm{UP}^O$.

As an immediate consequence we obtain:

Corollary 3.2.4 *The following holds for the oracle O constructed in Theorem 3.2.3.*

1. DisjNP^O *does not have* $\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-complete pairs.*
2. $\mathrm{NP}^O \cap \mathrm{coNP}^O$ *does not have* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete sets.*
3. UP^O *has* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete sets.*

Remark 3.2.5 *Without loss of generality, we may assume in the following proof that each* UP*-set is accepted by infinitely many* UP*-machines in the standard enumeration $M_1, M_2, \ldots$, i.e., more precisely, for each oracle D and each $A \in \mathrm{UP}^D$ there are infinitely many $i \in \mathbb{N}^+$ such that (i) on every input x the computation $M_i^D(x)$ has at most one accepting path and (ii) $L(M_i^D) = A$. Note that typical standard enumerations satisfy this property.*

Proof of Theorem 3.2.3 Choose some $C \subseteq \Sigma^*$ that is $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete for PSPACE and whose elements all have odd length. Let $e(0) = 2$ and $e(n+1) = 2^{2^{e(n)}}$ for $n \in \mathbb{N}$. Define the following sets for $p \in \mathbb{P}_{\geq 3}$ and a (possibly partial) oracle D.

$$\begin{aligned} A_p^D &= \{0^{e(p^k)} \mid k \geq 1 \text{ and there is an even } x \in D \text{ such that } |x| = e(p^k)\} \cup \overline{\{0^{e(p^k)} \mid k \geq 1\}} \\ B_p^D &= \{0^{e(p^k)} \mid k \geq 1 \text{ and there is an odd } x \in D \text{ such that } |x| = e(p^k)\} \end{aligned}$$

These sets are clearly in NP. Note that if for each $k \geq 1$ it holds

$$\exists \text{ an even } x \in D \cap \Sigma^{e(p^k)} \;\Leftrightarrow\; \neg\exists \text{ an odd } x \in D \cap \Sigma^{e(p^k)},$$

then $A_p^D = \overline{B_p^D}$ and hence $A_p^D \in \mathrm{NP}^D \cap \mathrm{coNP}^D$.

Note that throughout this proof we sometimes omit the oracles in the superscript, e.g., we write NP or A_p instead of NP^D or A_p^D. However, we do not do that in the "actual" proof but only when roughly explaining ideas in order to convey intuition.

Preview of the Construction On the one hand, the construction tries to prevent that $L(M_i)$ and $L(M_j)$ for $i \neq j$ are disjoint. If this is not possible, then M_i and M_j inherently accept disjoint sets. In this case, for a suitable $p \in \mathbb{P}_{\geq 3}$, the construction makes sure that A_p is in NP ∩ coNP and does not $\leq_{\mathrm{m}}^{\mathrm{p}}$-reduce to $(L(M_i), L(M_j))$. This prevents the existence of disjoint NP-pairs that are $\leq_{\mathrm{m}}^{\mathrm{pp},O}$-hard for $\mathrm{NP}^O \cap \mathrm{coNP}^O$.

On the other hand, the construction tries to prevent that M_i has the *uniqueness property*, i.e., for all x the computation $M_i(x)$ has at most one accepting path. If this is not possible, then M_i inherently has the uniqueness property, which enables us to show that $L(M_i)$ is in P relative to the final oracle.

During the oracle construction we maintain a growing collection of properties that we demand in the further construction. The collection is represented by a function t belonging to the set

$$\mathcal{T} = \big\{t\colon (\mathbb{N}^+)^2 \to \mathbb{P}_{\geq 3} \cup \{0,1\} \mid \mathrm{dom}(t) \text{ is finite, } \forall_{x,y \in (\mathbb{N}^+)^2}\ \big(t(x) = t(y) \in \mathbb{P}_{\geq 3}\big) \Rightarrow x = y, \text{ and } \forall_{(i,i) \in \mathrm{dom}(t)}\, t(i,i) \in \{0,1\}\big\}.$$

Note that the second of the three conditions can be equivalently expressed by saying that t is injective on the set $\{x \in (\mathbb{N}^+)^2 \mid t(x) > 1\}$. The last condition expresses that only pairs (i,j) with $i \neq j$ are mapped to some prime, i.e., $t^{-1}(\mathbb{P}_{\geq 3}) \subseteq \{(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+ \mid i \neq j\}$.

An oracle $w \in \Sigma^*$ is t-valid for some $t \in \mathcal{T}$ if the following statements hold:

V1: For all $(i,j) \in \mathrm{dom}(t)$ with $i \neq j$ and $t(i,j) = 0$ there exists z such that $M_i^w(z)$ and $M_j^w(z)$ definitely accept.
(meaning: for all extensions of the oracle the pair $(L(M_i), L(M_j))$ is not a disjoint pair.)

V2: For all odd primes $p \in \mathrm{ran}(t)$

1. $A_p^w \cap B_p^w = \emptyset$ and
2. for all $k \geq 1$ with $\forall_{z,|z|=e(p^k)}\, z < |w|$ there exists $x \in w$ with $|x| = e(p^k)$.

(meaning: relative to the final oracle it holds $A_p = \overline{B_p}$)

V3: For all $(i,j) \in \mathrm{dom}(t)$ with $i = j$ and $t(i,i) = 0$ there exists z such that $M_i^w(z)$ has at least two paths that definitely accept.
(meaning: for all extensions of the oracle M_i^v violates the uniqueness property)

V4: If $x < |w|$ and $|x|$ is odd, then $x \in w \Leftrightarrow x \in C$.
(meaning: w and C coincide for words of odd length)

V5: If $x \in w$ and $|x|$ is even, then there exists $n \geq 1$ such that $|x| = e(n)$.
(meaning: w does not contain any words of even length except for the words of length $e(n)$ for some n)

This definition directly implies the following claim.

Claim 3.2.6 *Let $t, t' \in \mathcal{T}$ such that t' is an extension of t. If w is t'-valid, then w is t-valid.*

Claim 3.2.7 *Let $t \in \mathcal{T}$ and $u \sqsubseteq v \sqsubseteq w$ be oracles such that u and w are t-valid. Then v is t-valid.*

Proof The statement holds as v inherits the properties V1 and V3 from u and the properties V2, V4, and V5 from w. □

Claim 3.2.8 *For every $t \in \mathcal{T}$ and every t-valid partial oracle w there exists $b \in \{0,1\}$ such that wb is t-valid. More precisely, for $z = |w|$ the following holds.*

1. *If $|z|$ is odd, then the partial oracle $wC(z)$ is t-valid.*
2. *If $|z|$ is even, then the following holds.*
   (a) *If $|z| = e(n)$ for some $n \in \mathbb{N}$ and there exists no word $x \in w$ of length $|z|$, then $w1$ is t-valid.*
   (b) *If (i) for all primes $p \in \mathrm{ran}(t)$ and all $k \in \mathbb{N}^+$ it holds $z \neq 1^{e(p^k)}$ or (ii) there exists a word $x \in w$ with $|x| = |z|$, then $w0$ is t-valid.*

Proof Clearly V1 and V3 are not affected by extending the oracle by one bit.

1. If $|z|$ is odd, then extending the oracle by one bit does not have any influence on both V2 and V5. Moreover, $wC(z)$ trivially satisfies V4.

2. If $|z|$ is even, then $w0$ and $w1$ trivially satisfy V4.

Let us prove statement 2a. The oracle $w1$ clearly satisfies V2.2. Let us assume that z satisfies the conditions occurring in statement 2a. As there exists no word $x \in w$ of length $|z|$, $w1$ satisfies V2.1. As $|z| = e(p^k)$, V5 is satisfied by $w1$ as well. This proves statement 2a.

Finally, we prove statement 2b. Clearly $A_p^{w0} \cap B_p^{w0} = \emptyset$ for all odd primes $p \in \mathrm{dom}(t)$ and thus $w0$ satisfies V2.1. Furthermore, $w0$ trivially satisfies V5. By the requirements made for z in statement 2b, $w0$ also satisfies V2.2, which completes the proof of statement 2b. □

Oracle construction: Let T be an injective enumeration of $(\mathbb{N}^+)^2 \cup \{(i,j,r) \in (\mathbb{N}^+)^3 \mid i \neq j\}$ in an order having the property that for all $i, j, r \in \mathbb{N}^+$ with $i \neq j$ the pair (i,j) appears earlier than the triple (i,j,r). The elements of T should be considered as tasks. Note that during the construction we will delete tasks from T. We proceed stepwise and let each step treat the first task that still is in the current task list T and afterwards remove this and possibly other tasks from T.

Let t_0 be the unique nowhere defined function in $\mathcal{T}$ and $w_0 = \varepsilon$, which is t_0-valid. Starting with $t = t_0$ and $w = w_0$, we successively consider the tasks in T and treat a task by extending the function t and the partial oracle w in an appropriate way: thus we construct a sequence of partially defined oracles $w_0 \sqsubsetneq w_1 \sqsubsetneq \cdots$ and a sequence $t_0, t_1, \ldots$ of functions from $\mathcal{T}$ such that for all i the oracle w_i is t_i-valid and t_{i+1} is an extension of t_i. 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.

The final oracle is $O = \bigcup_{i\in\mathbb{N}} w_i$. It is totally defined, since each step strictly extends the oracle.

We now describe step $s > 0$, which starts with some $t_{s-1} \in \mathcal{T}$ and a t_{s-1}-valid oracle w_{s-1} and treats the first task that still is in T choosing an extension $t_s \in \mathcal{T}$ of t_{s-1} and a t_s-valid $w_s \sqsupsetneq w_{s-1}$. Let us recall that each task is immediately deleted from T after it is treated. We study three cases depending on the form of the task that is treated in step s (it will be argued later that the construction described in the following is indeed possible).

- task (i, j) with $i \neq j$: Let $t' = t_{s-1} \cup \{(i, j) \mapsto 0\}$. If there exists a t'-valid partial oracle $v \sqsupsetneq w_{s-1}$, then let w_s be the minimal such oracle (with respect to the quasi-lexicographical order of finite words[2]), let $t_s = t'$, and remove all tasks $(i, j, \cdot)$ from T. Otherwise, choose $p \in \mathbb{P}_{\geq 3} - \operatorname{ran}(t_{s-1})$ such that $p > |w_{s-1}|$ and let $t_s = t_{s-1} \cup \{(i, j) \mapsto p\}$ and $w_s = wb$ for $b \in \{0, 1\}$ such that w_s is t_s-valid. Note that such a bit b exists by Claim 3.2.8, since w_{s-1} is t_s-valid by the choice of p.
  (meaning: force $L(M_i^O) \cap L(M_j^O) \neq \emptyset$ if possible; otherwise, choose a suitable prime p and make sure that $A_p = \overline{B_p}$ with respect to the final oracle; corresponds to V1 and V2 in the definition of t-valid)

- task (i, i): Let $t' = t_{s-1} \cup \{(i, i) \mapsto 0\}$. If there exists a t'-valid partial oracle $v \sqsupsetneq w_{s-1}$, then let w_s be the minimal such oracle and let $t_s = t'$. Otherwise, let $t_s = t_{s-1} \cup \{(i, i) \mapsto 1\}$ and $w_s = wb$ for $b \in \{0, 1\}$ such that w_s is t_s-valid. Note that such a bit b exists by Claim 3.2.8, since w_{s-1} is t_s-valid.
  (meaning: destroy the uniqueness property of M_i if possible; otherwise, define $t_s(i, i) = 1$, which indicates that M_i inherently has the uniqueness property; corresponds to V3 in the definition of t-valid)

- task (i, j, r) with $i \neq j$: It holds $t_{s-1}(i, j) = p \in \mathbb{P}_{\geq 3}$ (otherwise, the task (i, j, r) would have been removed from the task list in the step treating the task (i, j)). Let $t_s = t_{s-1}$ and choose a t_s-valid $w_s \sqsupsetneq w_{s-1}$ such that for a suitable 0^n at least one of the following two assertions holds.
  - $0^n \in A_p^{w_s}$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
  - $0^n \in B_p^{w_s}$, $F_r^{w_s}(0^n)$ is definite, and $M_j^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

  (meaning: F_r does not realize a reduction $A_p \leq_{\mathrm{m}}^{\mathrm{p}} (L(M_i), L(M_j))$ relative to the final oracle)

Claim 3.2.9 *For all $s \geq 1$, the construction of w_s and t_s in step s is possible.*

Proof For a contradiction, assume that the statement is wrong and choose the smallest step s where the claim fails. There are two cases:

1. Step s treats a task (i, j) for $i, j \in \mathbb{N}^+$: Hence $t_{s-1}(i, j)$ is not defined, since it can only be defined by the unique treatment of the task (i, j). Therefore, t' can be defined as specified,

[2]Recall that this order coincides with the order "$\leq$" for natural numbers when we identify finite words and natural numbers in the way described in the preliminaries.

which shows that the construction in step s is possible (cf. the descriptions of the tasks of this form above).

2. Step s treats a task (i,j,r) with $i \neq j$: Here $t_s = t_{s-1}$ and $t_s(i,j) = p \in \mathbb{P}_{\geq 3}$, since otherwise, the earlier task (i,j) would have removed (i,j,r). We argue that the choice of the specified t_s-valid w_s is possible, which shows that the construction in step s is possible and which contradicts the assumption.

Choose k large enough such that for $n = e(p^k)$

- it holds $n \geq n_0$ where n_0 is the number that Corollary 3.2.2 guarantees to exist when applied for the parameters i, j, and r,
- there does not exist a word of length $\geq n$ which the oracle w_{s-1} is defined for, and
- $e(p^k+1) > (n^r+r)^{i+j} + i + j$.

Choose the minimal t_s-valid $w' \sqsupseteq w_{s-1}$ that is defined for all words of length $< n$ and undefined for all words of length $\geq n$ (such a partially defined oracle w' exists by Claim 3.2.8). By Corollary 3.2.2 applied for i, j, r, n, and $D := w' \cup C$, there exist an even $x \in \Sigma^n$ and an odd $y \in \Sigma^n$ such that at least one of the statements 1–3 in Corollary 3.2.2 holds.

If statement 1 holds, then define w_s as the minimal partial oracle $\sqsupsetneq w'$ that satisfies V4, contains x, and is defined for all words of length $\leq (n^r+r)^{i+j} + i + j$. Thus in particular, $w_s = w' \cup \{x\} \cup (C \cap \Sigma^{\leq (n^r+r)^{i+j}+i+j})$ when interpreting w' and w_s as sets. Let us show that w_s is t_s-valid. By $e(p^k+1) > (n^r+r)^{i+j} + i + j$ and $w_s \cap \Sigma^n = \{x\}$, the oracle w_s satisfies V2. By definition, w_s satisfies V4 and by $|x| = n = e(p^k)$ it satisfies V5. The remaining conditions V1 and V3 are not affected by extending the oracle w' to w_s. Hence w_s is t_s-valid. Since $x \in w_s$, it holds $0^n \in A_p^{w_s}$. By statement 1 of Corollary 3.2.2, the computation $M_i^{D \cup \{x\}}(F_r^{D \cup \{x\}}(0^n))$ rejects. As w_s is defined for all words of length $\leq (n^r+r)^{i+j} + i + j$ and agrees with $D \cup \{x\}$ for all these words, the computation $F_r^{w_s}(0^n)$ is definite and the computation $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

Thus we have seen that if statement 1 holds, then the construction in step s is possible. For statement 2 this is shown analogously.

It suffices to prove that statement 3 cannot hold. Otherwise,

$$F_r^{D \cup \{x,y\}}(0^n) \in L(M_i^{D \cup \{x,y\}}) \cap L(M_j^{D \cup \{x,y\}}).$$

Consider the step s' that treats the task (i,j), i.e., s' is the minimal number for which $t_{s'}(i,j)$ is defined and it holds $t_{s'} = t_{s'-1} \cup \{(i,j) \mapsto p\}$. Thus we have $s' \leq s-1$ and $w_{s'-1} \sqsubsetneq w_{s'} \sqsubseteq w_{s-1} \sqsubseteq w'$. As w' is t_s-valid, it is $t_{s'-1}$-valid by Claim 3.2.6. Choose the minimal $v \sqsupsetneq w'$ that satisfies V4, that contains x and y, and that is defined for all words of length $\leq (n^r+r)^{i+j} + i + j$. Hence in particular, when interpreting w' and v as sets, we have that $v = w' \cup \{x,y\} \cup (C \cap \Sigma^{\leq (n^r+r)^{i+j}+i+j})$ and thus v and $D \cup \{x,y\}$ agree on all words of length $\leq (n^r+r)^{i+j} + i + j$, which shows that both $M_i^v(F_r^v(0^n))$ and $M_j^v(F_r^v(0^n))$ definitely accept. This proves that if v is $t_{s'-1}$-valid, then it is even t'-valid for $t' = t_{s'-1} \cup \{(i,j) \mapsto 0\}$. Let us show that v is $t_{s'-1}$-valid (and thus t'-valid). As $e(p^k+1) > (n^r+r)^{i+j} + i + j$ and $p \notin \operatorname{ran}(t_{s'-1})$, the oracle v satisfies V2. By definition, v satisfies V4 and by $|x| = |y| = n = e(p^k)$ it satisfies V5. The remaining conditions V1 and V3 are not affected by extending w' to v. Hence v is $t_{s'-1}$-valid and thus t'-valid. Therefore, step s' defines $t_{s'} = t'$, which leads to a contradiction as $t_s(i,j) = p \neq 0 = t'(i,j) = t_{s'}(i,j)$. This shows that statement 3 cannot hold.

Thus we have shown that the choice of a t_s-valid w_s with the properties required in step s of the above construction is possible, which contradicts the assumption and finishes the proof of Claim 3.2.9. □

Claim 3.2.10 DisjNP^O *does not contain pairs that are* $\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-hard for* $\mathrm{NP}^O \cap \mathrm{coNP}^O$.

Proof Assume there exists a pair $(L(M_i^O), L(M_j^O)) \in \mathrm{DisjNP}^O$ that is $\leq_{\mathrm{m}}^{\mathrm{pp},O}$-hard for $\mathrm{NP}^O \cap \mathrm{coNP}^O$. From $L(M_i^O) \cap L(M_j^O) = \emptyset$ it follows that for all s there is no z such that $M_i^{w_s}(z)$ and $M_j^{w_s}(z)$ definitely accept. Hence $t_s(i,j) \neq 0$ for all s for which $t_s(i,j)$ is defined. Let s be the step that treats the task (i,j). Thus by construction, for all $s' \geq s$ it holds $t_{s'}(i,j) = p$ for some $p \in \mathbb{P}_{\geq 3}$, which by V2 implies $A_p^O = \overline{B_p^O} \in \mathrm{NP}^O \cap \mathrm{coNP}^O$. Thus by assumption, there exists an $r \in \mathbb{N}^+$ such that $(A_p^O, B_p^O) \leq_{\mathrm{m}}^{\mathrm{pp},O} (L(M_i^O), L(M_j^O))$ via F_r^O. Let s' be the step that treats the task (i,j,r). This step makes sure that for a suitable 0^n at least one of the following statements holds:

- $0^n \in A_p^{w_{s'}}$, $F_r^{w_{s'}}(0^n)$ is definite, and $M_i^{w_{s'}}(F_r^{w_{s'}}(0^n))$ definitely rejects.
- $0^n \in B_p^{w_{s'}}$, $F_r^{w_{s'}}(0^n)$ is definite, and $M_j^{w_{s'}}(F_r^{w_{s'}}(0^n))$ definitely rejects.

Note that by the definition of the sets $A_p^{\cdot}$ and $B_p^{\cdot}$, if $0^n A_p^{w_{s'}}$ (resp., $0^n \in B_p^{w_{s'}}$), then $0^n \in A_p^v$ (resp., $0^n \in B_p^v$) for all (finite or infinite) extensions v of $w_{s'}$. Thus we obtain that (i) $0^n \in A_p^O$ and $M_i^O(F_r^O(0^n))$ rejects or (ii) $0^n \in B_p^O$ and $M_j^O(F_r^O(0^n))$ rejects, which contradicts the choice of r and completes the proof. □

The proof of the following claim is based on a proof by Rackoff [Rac82, Theorem 4].

Claim 3.2.11 $\mathrm{P}^O = \mathrm{UP}^O$.

Proof It is known that $\mathrm{P}^{O'} \subseteq \mathrm{UP}^{O'}$ for all oracles O'. We prove $\mathrm{UP}^O \subseteq \mathrm{P}^O$. Let $L \in \mathrm{UP}^O$ and choose $i \in \mathbb{N}^+$ such that $L = L(M_i^O)$ and M_i^O has the uniqueness property, i.e., for all $x \in \Sigma^*$ the computation $M_i^O(x)$ has at most one accepting path. Such a number i exists according to Remark 3.2.5. Moreover, choose the smallest s such that $t_s(i,i)$ is defined and note that $t_s(i,i) = 1$ (otherwise, it would hold $t_s(i,i) = 0$ and thus by construction, M_i^O would not have the uniqueness property).

Let $x \in \Sigma^*$. If there exists an oracle D such that P is an accepting path of $M_i^D(x)$, then we call P a *potential accepting path of* $M_i(x)$.

For sets $Q, U, W, W' \subseteq \Sigma^*$ with $W \cup W' \subseteq U$ and $W \cap W' = \emptyset$ we say that P *respects* (Q, U, W, W') if it answers *yes* to all questions in $C \cup Q \cup W$, *no* to all questions in W', and *no* to all questions not in $C \cup Q \cup U$. Note that by the definition of potential accepting paths, queries in $U - (W \cup W')$ are answered consistently on the path P. Moreover, P^{all} (resp., P^{yes} or P^{no}) denotes the set of all (resp., all positively answered or all negatively answered) queries of P.

Consider the following algorithm that we will prove to decide L.

```
1. Input: x ∈ Σ*
2. Let m = |x|.
3. If m < 4, m^i + i > 2^m, or w_{s-1} is defined for 0^{log m}:
4.    If x ∈ L, then Accept, else Reject.
5. Let n be the unique number with e(n − 1) ≤ log m < e(n).
6. Let Q = {q ∈ 0 | |q| even and |q| < e(n)}.
7. If n = p^k for some k ≥ 1 and some odd prime p ∈ ran(t_{s-1}):
8.    Let U = {z ∈ Σ* | |z| = e(n) and z odd} and W = W' = ∅.
9.    If SEARCH returns True, then Accept.
10.   Let U = {z ∈ Σ* | |z| = e(n) and z even} and W = W' = ∅.
11.   If SEARCH returns True, then Accept.
12.   Reject.
```

```
13. If n ≠ p^k for all k ≥ 1 and all odd primes p ∈ ran(t_{s-1}):
14.    Let U = {z ∈ Σ* | |z| = e(n)} and W = W' = ∅.
15.    If SEARCH returns True, then Accept.
16.    Reject.

17. subroutine SEARCH
18.    For j = 0 to 4(m^i + i):
19.       If there is no potential accepting path of M_i(x) respecting (Q,U,W,W'),
          then return False, else let P be such a path.
20.       For each z ∈ P^all with |z| = e(n):
21.          Ask whether z ∈ O.
22.          If z ∈ O − U, then return False.
23.          If z ∈ O ∩ U, then insert z into W.
24.          If z ∈ O̅ ∩ U, then insert z into W'.
25.       If P still respects (Q,U,W,W'), then return True.
26.    Return False.
```

Note that the condition $m < 4$ in line 3 is necessary as for $m < 4$ there does not exist an $n \in \mathbb{N}^+$ with $e(n-1) \leq \log m$, which is introduced in line 5.

We argue that the algorithm can be implemented by a polynomial-time oracle Turing machine with oracle O and observe that it suffices to argue for the lines 6, 7–12, 13–16, and 17–26. So we consider the algorithm on some input $x \in \Sigma^*$ of length $m \in \mathbb{N}$ and assume that the algorithm does not terminate in line 4, but enters line 6.

Line 6: If this line is executed, then n has been defined and it holds $e(n-1) \leq \log m$. Recall that due to V4 and V5 in the definition of t-valid each word in $O - C$ is of length $e(j)$ for some j. Thus the set Q consists of all words in O that are of length $e(j)$ for some $j \in [0, n-1]$. From $e(n-1) \leq \log m$ we obtain $|\bigcup_{j=0}^{n-1} \Sigma^{e(j)}| \leq \sum_{j=0}^{n-1} 2^{e(j)} < 2^{e(n-1)+1} \leq 2m$, which shows that a polynomial-time oracle Turing machine with oracle O can ask "$q \in O$?" for all $q \in \bigcup_{j=0}^{n-1} \Sigma^{e(j)}$ and thus compute Q.

Lines 7–12 and 13–16: Note that we introduce the set U in the lines 8, 10, and 14 only for better readability. These sets never have to be computed or stored explicitly, since it can be easily checked whether some word is in the respective set or not.

Hence it remains to argue for the lines 17–26, i.e., for subroutine SEARCH. Note that all words whose membership in C, Q, U, W, and W' is tested in these lines are of length at most $m^i + i$ (all these words are queried by some potential accepting path P of $M_i(x)$ which is of length at most $m^i + i$). Thus testing the membership of words in Q, U, W, and W' in lines 19, 22, 23, 24, and 25 is possible in polynomial time in $m = |x|$ without oracle access (as for W and W' observe that $|W|$ and $|W'|$ are bounded by $(4(m^i+i)+1) \cdot (m^i+i)$ throughout the execution of the algorithm as there are $4(m^i+i)+1$ iterations of the loop starting in line 18 and each path P considered in the loop is of length $\leq m^i+i$). Due to that and $C \in \mathrm{PSPACE}$, we can determine in polynomial space in $|x|$ without oracle access (whether there exists) a potential accepting path of $M_i(x)$ that respects (Q, U, W, W'). As $\mathrm{PSPACE} \subseteq \mathrm{P}^C \subseteq \mathrm{P}^O$ and $\mathrm{FPSPACE} \subseteq \mathrm{FP}^C \subseteq \mathrm{FP}^O$, the subroutine SEARCH only requires polynomial time in $|x|$ when having access to the oracle O, which completes the proof of the assertion that the above algorithm can be implemented by a deterministic polynomial-time oracle Turing machine with access to the oracle O.

Before discussing the question of which language the algorithm accepts, we make some general observations.

$$\text{Whenever the algorithm executes some line } i \geq 6, \text{ it holds } m^i + i \leq 2^m < 2^{2^{\log m + 1}} \leq 2^{2^{e(n)}} = e(n+1). \tag{3.1}$$

Recall that O consists of C and elements of length $e(j)$ for some $j \in \mathbb{N}$. Due to that and (3.1) we obtain the following statement.

$$\text{Whenever the algorithm executes some line } i \geq 6, \text{ it holds } O \cap \Sigma^{\leq m^i+i} = \left(C \cap \Sigma^{\leq m^i+i}\right) \cup Q \cup \left(O \cap \Sigma^{e(n)}\right). \tag{3.2}$$

As a last observation, due to the lines 23–24 and the choice of U we have the following.

$$\text{Throughout the execution of the subroutine SEARCH it always holds } W \cup W' \subseteq U \subseteq \Sigma^{e(n)} \wedge W \subseteq O \wedge W' \subseteq \overline{O}. \tag{3.3}$$

Let us now prove that the algorithm indeed accepts L. We start with showing the following implication: if the algorithm accepts some input x, then $x \in L$. This is true if the algorithm accepts in line 4. So we now consider the case that it accepts in one of the lines 9, 11, and 15. Then in the corresponding line the respective call of the subroutine SEARCH returns True. Let us have a closer look at this call of SEARCH in the following.

Consider the iteration of the loop 18–25 that in line 25 returns True. Hence, when the latter line is executed, it holds that the current potential accepting path P respects (Q, U, W, W'). Let q be some query asked by P and note that thus $|q| \leq m^i + i$. We study several cases:

- If $|q| < e(n)$, then the answer to q on the path P is $(C \cup Q)(q)$ (by the definition of "respects" and (3.3)) and it holds $(C \cup Q)(q) = O(q)$ (by (3.2)).
- If $|q| > e(n)$, then the answer to q on the path P is $C(q)$ (by the definition of "respects", $Q \subseteq \Sigma^{\leq e(n-1)}$, and (3.3)) and it holds $C(q) = O(q)$ (by $Q \subseteq \Sigma^{\leq e(n-1)}$ and (3.2)).
- If $|q| = e(n)$ and $q \in W$, then the answer to q on the path P is $1 = O(q)$ (by the definition of "respects" and (3.3)).
- If $|q| = e(n)$ and $q \in W'$, then the answer to q on the path P is $0 = O(q)$ (by the definition of "respects" and (3.3)).
- If $|q| = e(n)$ and $q \in \overline{O} - U$, then by the definition of "respects" and $C \cup Q \subseteq O$, the answer to q on the path P is 0, which equals $O(q)$.

By the lines 22–24 and since the algorithm returns True in line 25, we have $P^{\text{all}} \cap \Sigma^{e(n)} \subseteq W \cup W' \cup (\overline{O} - U)$, which shows that the cases $(|q| = e(n) \wedge q \in U - (W \cup W'))$ and $(|q| = e(n) \wedge q \in O - U)$ are impossible (recall that $W \cup W' \subseteq U$ by (3.3)). Hence in the aforementioned execution of line 25, P is an accepting path of $M_i^O(x)$, which implies $x \in L$. This shows that if the algorithm accepts x, then $x \in L$.

It remains to argue that if $x \in L$, then the algorithm accepts x. Let $x \in L$. If the algorithm on input x terminates in line 4, then the algorithm accepts. Thus we assume from now on that $m^i + i \leq 2^m$ and w_{s-1} is undefined for all words of length $\geq \log m$, where the latter and the choice of n in line 5 imply in particular that w_{s-1} is undefined for all words of length $e(n)$. We consider two cases:

Case 1: $4(m^i + i) \geq 2^{e(n)}$.

Assume that the algorithm does not accept, i.e., it rejects. We show that this leads to a contradiction. The assumption that the algorithm does not stop in line 4 implies that it stops in one of the lines 12 or 16. Note that if the algorithm stops in line 12, then $0^{e(n)} \notin A_p^O$ or $0^{e(n)} \notin B_p^O$ since $p \in \text{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3}$ and hence $A_p^O \cap B_p^O = \emptyset$ by V2.1. We have to consider the following cases.

Case 1a: The algorithm stops in line 12 and $0^{e(n)} \notin A_p^O$. Here we continue the argumentation by choosing $U = \{z \in \mathbb{N} \mid |z| = e(n) \text{ and } z \text{ odd}\}$ and having a closer look at the call of SEARCH in line 9, which returns False.

Case 1b: The algorithm stops in line 12 and $0^{e(n)} \notin B_p^O$. Here we continue the argumentation by choosing $U = \{z \in \mathbb{N} \mid |z| = e(n) \text{ and } z \text{ even}\}$ and having a closer look at the call of SEARCH in line 11, which returns False.

Case 1c: The algorithm stops in line 16. Here we continue the argumentation by choosing $U = \{z \in \mathbb{N} \mid |z| = e(n)\}$ and having a closer look at the call of SEARCH in line 15, which returns False.

We argue for the Cases 1a, 1b, and 1c in parallel. Note that in each case it holds $O \cap \Sigma^{e(n)} \subseteq U$. As $x \in L$, the computation $M_i^O(x)$ has an accepting path P'. Then P' is a potential accepting path of $M_i(x)$. Let q be an arbitrary query asked on the path P'. Thus $|q| \leq m^i + i$. Studying several cases, we show that P' respects (Q, U, W, W') whenever we reach line 19.

- If $q \in C \cup Q \cup W$, then by $C \cup Q \subseteq O$ and (3.3), it holds $O(q) = 1$ and hence the answer to the query q on path P' is 1.
- If $q \in W'$, then (3.3) implies $O(q) = 0$ and thus the answer to the query q on path P' is 0.
- If $q \notin C \cup Q \cup U$, then by (3.2) and $O \cap \Sigma^{e(n)} \subseteq U$, it holds $O(q) = 0$ and thus the answer to the query q on path P' is 0.

Hence the considered call of SEARCH cannot return False in line 19. Moreover, by $O \cap \Sigma^{e(n)} \subseteq U$, it cannot return False in line 22. Thus the considered call of SEARCH returns False in line 26. Then in particular, the loop 18–25 is executed exactly $4(m^i + i) + 1$ times and in each execution of the loop 18–25 the value of $|W \cup W'|$ is increased by at least 1 (otherwise, the path P chosen in line 19 still respects (Q, U, W, W') in line 25 and the subroutine returns True). Hence, when reaching line 26 it holds $|W \cup W'| > 4(m^i + i) \geq 2^{e(n)} = |\Sigma^{e(n)}|$, in contradiction to $W \cup W' \subseteq \Sigma^{e(n)}$, which holds according to (3.3).

Case 2: $4(m^i + i) < 2^{e(n)}$.

Define the following predicate.

$$\begin{aligned}&\textit{All potential accepting paths } P_1, P_2 \textit{ of } M_i(x) \textit{ that respect } (Q, U, W, W') \textit{ and}\\ &\textit{that satisfy } P_1^{\text{all}} \cap (U - (W \cup W')) \neq \emptyset \textit{ and } P_2^{\text{all}} \cap (U - (W \cup W')) \neq \emptyset \textit{ have a}\\ &\textit{query from } U - (W \cup W') \textit{ in common, i.e., } P_1^{\text{all}} \cap P_2^{\text{all}} \cap (U - (W \cup W')) \neq \emptyset.\end{aligned} \tag{3.4}$$

We prove the following assertions.

$$\begin{aligned}&\textit{If } n = p^k \textit{ for } p \in \operatorname{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3} \textit{ and } k \geq 1, U = \{z \mid |z| = e(n) \text{ and } z \text{ odd}\}, W \subseteq O \cap U,\\ &W' \subseteq \overline{O} \cap U, \textit{ and } 4(m^i + i) < 2^{e(n)}, \textit{ then (3.4) holds.}\end{aligned} \tag{3.5}$$

$$\begin{aligned}&\textit{If } n = p^k \textit{ for } p \in \operatorname{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3} \textit{ and } k \geq 1, U = \{z \mid |z| = e(n) \text{ and } z \text{ even}\}, W \subseteq O \cap U,\\ &W' \subseteq \overline{O} \cap U, \textit{ and } 4(m^i + i) < 2^{e(n)}, \textit{ then (3.4) holds.}\end{aligned} \tag{3.6}$$

$$\begin{aligned}&\textit{If } n \neq p^k \textit{ for all } p \in \operatorname{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3} \textit{ and all } k \geq 1, U = \{z \mid |z| = e(n)\}, W \subseteq O \cap U,\\ &W' \subseteq \overline{O} \cap U, \textit{ and } 4(m^i + i) < 2^{e(n)}, \textit{ then (3.4) holds.}\end{aligned} \tag{3.7}$$

By symmetry, if suffices to prove (3.5) and (3.7). As (3.7) can be proven in almost the same way as (3.5) (as a matter of fact, in an even simpler way), it is sufficient to prove (3.5) only. Suppose there exist potential accepting paths P_1, P_2 of $M_i(x)$ that respect (Q, U, W, W'), that satisfy $P_1^{\text{all}} \cap (U - (W \cup W')) \neq \emptyset$ and $P_2^{\text{all}} \cap (U - (W \cup W')) \neq \emptyset$, and that do not have a query from $U - (W \cup W')$ in common. We will lead this assumption to a contradiction by proving that then the construction would have destroyed the uniqueness property of M_i and thus chosen $t_s(i, i)$ to be 0.

First observe that by the above assumption, the paths P_1 and P_2 are distinct.

Let $Y = (P_1^{\text{yes}} \cup P_2^{\text{yes}}) \cap \Sigma^{\geq e(n)} = (P_1^{\text{yes}} \cup P_2^{\text{yes}}) \cap \Sigma^{[e(n),m^i+i]}$ and $N = (P_1^{\text{no}} \cup P_2^{\text{no}}) \cap \Sigma^{\geq e(n)} = (P_1^{\text{no}} \cup P_2^{\text{no}}) \cap \Sigma^{[e(n),m^i+i]}$. As P_1 and P_2 are potential accepting paths of $M_i(x)$ it holds $|N| \leq \ell(N) \leq 2 \cdot (m^i + i)$. As P_1 and P_2 respect (Q, U, W, W'), $W \subseteq U$, and $Q \subseteq \Sigma^{<e(n)}$, we have $Y \subseteq U \cup C$.

Let us argue for $Y \cap N = \emptyset$. Assume there exists some $q \in Y \cap N$. Hence $|q| \geq e(n)$. If $|q| > e(n)$, then $q \in Y \subseteq U \cup C$ implies $q \in C$, which contradicts $q \in N$ since both paths respect (Q, U, W, W'). So we assume $|q| = e(n)$, which implies $q \notin C$. We obtain:

- If $q \notin U$, then by $Y \subseteq U \cup C$ and $q \notin C$, we obtain $q \notin Y$, a contradiction.
- If $q \in W$, then $q \notin P_1^{\text{no}}$ and $q \notin P_2^{\text{no}}$ since both P_1 and P_2 respect (Q, U, W, W'). This contradicts $q \in N$.
- If $q \in W'$, then $q \notin P_1^{\text{yes}}$ and $q \notin P_2^{\text{yes}}$ since both P_1 and P_2 respect (Q, U, W, W'). This contradicts $q \in Y$.
- Assume $q \in U - (W \cup W')$. As both P_1 and P_2 are potential accepting paths of $M_i(x)$, it holds $q \notin P_1^{\text{yes}} \cap P_1^{\text{no}}$ and $q \notin P_2^{\text{yes}} \cap P_2^{\text{no}}$. Due to that and $q \in Y \cap N$ it holds $q \in P_1^{\text{yes}} \cap P_2^{\text{no}}$ or $q \in P_1^{\text{no}} \cap P_2^{\text{yes}}$. Hence P_1 and P_2 have a query from $U - (W \cup W')$ in common, which contradicts the assumption.

This shows $Y \cap N = \emptyset$.

Let $u \sqsupseteq w_{s-1}$ such that $u(z) = O(z)$ for all words z with $|z| < e(n)$ and u is undefined for all other words (recall that w_{s-1} is undefined for all words of length $e(n)$ as was observed in the last paragraph before we start to consider Case 1). Note $w_{s-1} \sqsubseteq u \sqsubseteq w_{s'}$ for a sufficiently large $s' > s - 1$. By construction, $w_{s'}$ is $t_{s'}$-valid and thus by Claim 3.2.6, it also is t_{s-1}-valid. Hence Claim 3.2.7 yields that u is t_{s-1}-valid. Note that it holds $u = O \cap \{w \in \Sigma^* \mid w < |u|\}$ (i.e., u is a prefix of O).

Consider the minimal $v \sqsupsetneq u$ that satisfies V4, that contains all words in Y, that contains at least one word from $U - N$ (such a word exists since $|N| \leq 2(m^i + i)$, $|U| = 2^{e(n)-1}$, and $4(m^i + i) < 2^{e(n)}$, where the latter condition holds by the assumption we have made in the beginning of Case 2), and that is defined for all words of length $\leq \max(m^i + i, e(n))$. The non-emptiness of $U - N$ is the reason for the distinction of the Cases 1 and 2.

Let us prove that v is t_{s-1}-valid. The oracle v satisfies V2.1 and V5, since u satisfies V2.1 and V5 and the words of even length in $v - u$ are all in $U = \{z \mid |z| = e(n) \text{ and } z \text{ odd}\}$ (recall $Y \subseteq U \cup C$). It also satisfies V2.2, since u satisfies V2.2, v contains at least one word from $U - N$, and it holds $e(n+1) > \max(m^i + i, e(n))$ by (3.1). Finally, V1 and V3 are not affected by extending the oracle. Thus v is t_{s-1}-valid.

We have the following cases for queries q on P_1 and P_2 and their respective answers:

- If $|q| < e(n)$, then as P_1 and P_2 respect (Q, U, W, W'), the answer to the query q on path P_1 or P_2 is $(C \cup Q)(q) \overset{(3.2)}{=} O(q) = u(q) = v(q)$.
- If $|q| \geq e(n)$ and $q \in Y$, then by the choice of Y and v, the answer is $1 = v(q)$.
- If $|q| \geq e(n)$ and $q \in N$, then by the choice of N as well as $Y \cap N = \emptyset$, the answer to queries q on the paths P_1 and P_2 is 0 at all times. As $v \cap N = \emptyset$ (it holds that (i) $Y \cap N = \emptyset$ and (ii) both P_1 and P_2 respect (Q, U, W, W'), which implies $N \cap C = \emptyset$), we have $v(q) = 0$.

This shows that P_1 and P_2 are two different accepting paths of the computation $M_i^v(x)$. Both paths are definitely accepting, since v is defined for all words of length $\leq m^i + i$. Thus v is

t'-valid for $t' = t_{s-1} \cup \{(i,i) \mapsto 0\}$. Hence step s defines $t_s = t'$, which contradicts $t_s(i,i) = 1$. This proves (3.5).

We continue to argue that the algorithm accepts x. For that purpose we study two subcases.

Case 2a: Assume $n = p^k$ for some $p \in \mathrm{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3}$ and $k \geq 1$. Then $A_p^O \cap B_p^O = \emptyset$ due to V2.1. Without loss of generality, $0^{e(n)} \notin A_p^O$ (otherwise, $0^{e(n)} \notin B_p^O$ and it can be argued symmetrically), i.e., O does not contain an even word of length $e(n)$. Consider the lines 8 and 9 and the respective call of the subroutine SEARCH. This means $U = \{z \in \mathbb{N} \mid |z| = e(n) \text{ and } z \text{ odd}\}$. Hence $O - U$ does not contain any words of length $e(n)$ and thus the aforementioned call of SEARCH does not return False in line 22. Since $x \in L$, there exists an accepting path $\widetilde{P}$ of $M_i^O(x)$.

Let us prove next that for all $W \subseteq O \cap U$ and $W' \subseteq \overline{O} \cap U$ the (potential accepting) path $\widetilde{P}$ respects (Q, U, W, W'). Let $q \in \widetilde{P}^{\mathrm{all}}$, $W \subseteq O \cap U$, and $W' \subseteq \overline{O} \cap U$.

- If $q \in C \cup Q \cup W$, then the answer to q on $\widetilde{P}$ is *yes*, since $\widetilde{P}$ is a path of the computation $M_i^O(x)$ and $C \cup Q \cup W \subseteq O$.
- If $q \in W'$, then the answer to q on $\widetilde{P}$ is *no*, since $\widetilde{P}$ is a path of the computation $M_i^O(x)$ and $W' \subseteq \overline{O}$.
- Assume $q \notin C \cup Q \cup U$. As we observed above that $(O - U) \cap \Sigma^{e(n)} = \emptyset$, we obtain by (3.2) that $O \cap \Sigma^{\leq m^i + i} \subseteq C \cup Q \cup U$ and hence $q \notin O$. Thus the answer to q on the path $\widetilde{P}$ is *no*.

During the execution of SEARCH it always holds that $W \subseteq O \cap U$ and $W' \subseteq \overline{O} \cap U$ (cf. (3.3)). Thus each time we reach line 19 it holds that $\widetilde{P}$ is a potential accepting path respecting (Q, U, W, W'). Hence in each iteration of the loop 18–25 line 19 does not return False. Let us consider an arbitrary iteration of the loop. In line 19 some potential accepting path P_1 that respects (Q, U, W, W') is chosen.

- If $P_1^{\mathrm{all}} \cap (U - (W \cup W')) = \emptyset$ during the execution of line 19, then P_1 still respects (Q, U, W, W') when reaching line 25 and hence SEARCH returns True.
- Otherwise, we have $P_1^{\mathrm{all}} \cap (U - (W \cup W')) \neq \emptyset$ during the execution of line 19 and by (3.5), for each potential accepting path P_2 that respects (Q, U, W, W') and that satisfies $P_2^{\mathrm{all}} \cap (U - (W \cup W')) \neq \emptyset$ the paths P_1 and P_2 have a query $\in U - (W \cup W')$ in common. This query is inserted into W or W' in one of the lines 23 and 24, and thus $|P_2^{\mathrm{all}} \cap (U - (W \cup W'))|$ is decreased by at least 1 in in the iteration chosen above.

Therefore, if SEARCH does not return True within $m^i + i$ iterations of the loop 18–25, then after this number of iterations all potential accepting paths P_2 that respect (Q, U, W, W') satisfy $P_2^{\mathrm{all}} \cap (U - (W \cup W')) = \emptyset$. Moreover, there exists at least one such path, namely $\widetilde{P}$, which is the reason why the next iteration returns True in line 25. This implies that the loop returns True within $m^i + i + 1$ iterations, which shows that the algorithm accepts in line 9.

Case 2b: Assume $n \neq p^k$ for all $p \in \mathrm{ran}(t_{s-1}) \cap \mathbb{P}_{\geq 3}$ and all $k \geq 1$. Consider the lines 14 and 15 (here $U = \Sigma^{e(n)}$). Due to the choice of U, the subroutine SEARCH does not return False in line 22. Since $x \in L$, there exists an accepting path $\widetilde{P}$ of $M_i^O(x)$.

In a similar, but even simpler way than in Case 2a it can be proven that for all $W \subseteq O \cap U$ and $W' \subseteq \overline{O} \cap U$ the (potential accepting) path $\widetilde{P}$ respects (Q, U, W, W').

In order to complete the proof that also in the present case the algorithm accepts, it suffices to copy the corresponding part of the proof in Case 2a and replace (3.5) with (3.7). This completes the proof of Claim 3.2.11. □

Now the proof of Theorem 3.2.3 is complete. □

3.3 DisjNP, UP, NP ∩ coNP, and ¬SAT Relative to an Oracle

As the title suggests, in this section we construct an oracle relative to which DisjNP ∧ UP ∧ NP ∩ coNP ∧ ¬SAT holds. As we have already motivated and discussed this oracle in the introduction, we only add one more aspect here: among others, the oracle shows that there are no relativizable proofs for the implication NP∩coNP ⇒ SAT. Recently Fabian Egidy, Anton Ehrmanntraut, and Christian Glaßer [unpublished, private communication] have constructed several oracles among which there is one illustrating that neither the converse of the aforementioned implication can be proven solely exploiting relativizable proof techniques, i.e., it holds SAT ∧ ¬NP ∩ coNP relative to their oracle[3]. Thus the conjectures SAT and NP ∩ coNP are independent in the sense that it does not hold that one of the conjectures implies the other one relative to all oracles.

Theorem 3.3.1 *There exists an oracle O such that the following statements hold:*

1. DisjNP^O *does not contain pairs that are* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-hard for* $\mathrm{UP}^O \cap \mathrm{coUP}^O$.
2. *Each non-empty* $L \in \mathrm{NP}^O$ *has* P^O*-optimal proof systems.*
3. UP^O *does not contain* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problems.*

This implies the subsequent corollary.

Corollary 3.3.2 *There exists an oracle O such that the following statements hold:*

1. DisjNP^O *does not contain* $\leq_{\mathrm{m}}^{\mathrm{pp},O}$*-complete pairs.*
2. *Each non-empty* $L \in \mathrm{NP}^O$ *has* P^O*-optimal proof systems.*
3. UP^O *does not contain* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problems.*
4. $\mathrm{NP}^O \cap \mathrm{coNP}^O$ *does not contain* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problems.*

Remark 3.3.3 *Without loss of generality, we may assume in the following proof that each* UP*-set is accepted by infinitely many* UP*-machines in the standard enumeration $M_1, M_2, \ldots$, i.e., more precisely, for each oracle D and each $A \in \mathrm{UP}^D$ there are infinitely many $i \in \mathbb{N}^+$ such that (i) on every input x the computation $M_i^D(x)$ has at most one accepting path and (ii) $L(M_i^D) = A$. Note that typical standard enumerations satisfy this property.*

Proof of Theorem 3.3.1 Let D be a (possibly partial) oracle, $p \in \mathbb{P}_{\equiv 3} := \mathbb{P} \cap \{4k+3 \mid k \in \mathbb{N}\}$, and $q \in \mathbb{P}_{\equiv 1} := \mathbb{P} \cap \{4k+1 \mid k \in \mathbb{N}\}$ (note that $\mathbb{P}_{\equiv 3}$ and $\mathbb{P}_{\equiv 1}$ are both infinite sets). We define

$$\begin{aligned} A_p^D &:= \{0^{p^k} \mid k \in \mathbb{N}^+, \exists_{x \in \Sigma^{p^k}} x \in D \text{ and } x \text{ odd}\} \cup \overline{\{0^{p^k} \mid k \in \mathbb{N}^+\}} \\ B_p^D &:= \{0^{p^k} \mid k \in \mathbb{N}^+, \exists_{x \in \Sigma^{p^k}} x \in D \text{ and } x \text{ even}\} \\ C_q^D &:= \{0^{q^k} \mid k \in \mathbb{N}^+, \exists_{x \in \Sigma^{q^k}} x \in D\} \end{aligned}$$

Note that $A_p^D, B_p^D \in \mathrm{NP}^D$ and $A_p^D = \overline{B_p^D}$ if $|\Sigma^{p^k} \cap D| = 1$ for each $k \in \mathbb{N}^+$. In that case $A_p^D \in \mathrm{UP}^D \cap \mathrm{coUP}^D$. Moreover, $C_q^D \in \mathrm{UP}^D$ if $|\Sigma^{q^k} \cap D| \leq 1$ for each $k \in \mathbb{N}^+$.

Note that throughout this proof we sometimes omit the oracles in the superscript, e.g., we write NP or A_p instead of NP^D or A_p^D. However, we do not do that in the "actual" proof but only when explaining ideas in a loose way in order to convey the intuition behind the occasionally rather technical arguments.

[3] Indeed, even DisjCoNP ∧ ¬NP ∩ coNP holds relative to the oracle.

For $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ we write $c(i,x,y) := \langle 0^i, x, 0^{|x|^i+i}, 0^{|x|^i+i}, y, y\rangle$. Note that $|c(i,x,y)|$ is even, polynomially bounded in $|i|+|x|+|y|$, and by the properties of the pairing function $\langle\cdot\rangle$,

$$\forall_{i\in\mathbb{N}^+,x,y\in\mathbb{N}}\ |c(i,x,y)| > 4\cdot\max(|x|^i+i,|y|). \tag{3.8}$$

Claim 3.3.4 ([DG19]) *Let $w \in \Sigma^*$, $i \in \mathbb{N}^+$, and $x \in \mathbb{N}$. If $c(i,x,y) \le |w|$ for some $y \in \mathbb{N}$, then the following holds.*

1. *$F_i^w(x)$ is definite and $F_i^w(x) < |w|$.*
2. *For all $v \sqsupseteq w$, it holds $(F_i^w(x) \in \mathcal{CAN}^w \Leftrightarrow F_i^w(x) \in \mathcal{CAN}^v)$.*

Proof As the running time of $F_i^w(x)$ is bounded by $|x|^i + i < |c(i,x,y)| < c(i,x,y) \le |w|$, the computation $F_i^w(x)$ is definite and its output is $< |w|$. Hence 1 holds. Consider 2. It suffices to show that $\mathcal{CAN}^v(q) = \mathcal{CAN}^w(q)$ for all $q < |w|$ and all $v \sqsupseteq w$. This holds by Lemma 3.1.1. □

Preview of the Construction We sketch some of the basic ideas our construction uses.

1. For all positive and distinct i and j the construction tries to achieve that $(L(M_i), L(M_j))$ is not a disjoint NP-pair. If this is not possible, then $(L(M_i), L(M_j))$ inherently is a disjoint NP-pair. Once we know this, we choose some prime $p \in \mathbb{P}_{\equiv 3}$, ensure $A_p, B_p \in \mathrm{UP}$, $A_p = \overline{B_p}$, and thus $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ in the further construction, and diagonalize against all FP-functions such that A_p is not $\le_{\mathrm{m}}^{\mathrm{p}}$-reducible to $(L(M_i), L(M_j))$.

2. For all $i \ge 1$ the construction intends to make sure that F_i is not a proof system for $\mathcal{CAN}$. If this is not possible, then F_i inherently is a proof system for $\mathcal{CAN}$. Then we start to encode the values of F_i into the oracle, which will allow us to define a proof system that simulates all others relative to the oracle. However, it is important to also allow encodings for functions that are not known to be proof systems for $\mathcal{CAN}$ yet. Thus the final oracle will also contain encodings for functions that are not proof systems for $\mathcal{CAN}$.

3. For all $i \ge 1$ the construction tries to ensure that M_i is not a UP-machine. In case this is impossible, we know that M_i inherently is a UP-machine, which enables us to diagonalize against all FP-functions making sure that C_q for some $q \in \mathbb{P}_{\equiv 1}$ that we choose is not reducible to $L(M_i)$.

During the construction we maintain a growing collection of requirements that is specified by a partial function belonging to the set

$$\mathcal{T} = \Big\{ t\colon \mathbb{N}^+ \cup (\mathbb{N}^+)^2 \to \mathbb{Z} \mid \mathrm{dom}(t) \text{ is finite, } t \text{ is injective on its support,}$$
$$\bullet\ t(\mathbb{N}^+) \subseteq \{0\} \cup \mathbb{N}^+$$
$$\bullet\ t(\{(i,i) \mid i \in \mathbb{N}^+\}) \subseteq \{0\} \cup \{-q \mid q \in \mathbb{P}_{\equiv 1}\}$$
$$\bullet\ t(\{(i,j) \in (\mathbb{N}^+)^2 \mid i \ne j\}) \subseteq \{0\} \cup \{-p \mid p \in \mathbb{P}_{\equiv 3}\}\Big\}.$$

A partial oracle $w \in \Sigma^*$ is called t-valid for $t \in \mathcal{T}$ if it satisfies the following requirements.

V1 For all $i \in \mathbb{N}^+$ and all $x, y \in \mathbb{N}$, if $c(i,x,y) \in w$, then $F_i^w(x) = y \in \mathcal{CAN}^w$.
(meaning: if the oracle contains the codeword $c(i,x,y)$, then $F_i^w(x)$ outputs y and $y \in \mathcal{CAN}^w$; hence $c(i,x,y) \in w$ is a proof for $y \in \mathcal{CAN}^w$.)

V2 For all distinct $i, j \in \mathbb{N}^+$ with $t(i,j) = 0$ there exists x such that $M_i^w(x)$ and $M_j^w(x)$ definitely accept.
(meaning: for every extension of the oracle, $(L(M_i), L(M_j))$ is not a disjoint NP-pair.)

V3 For all distinct $i, j \in \mathbb{N}^+$ with $t(i,j) = -p$ for some $p \in \mathbb{P}_{\equiv 3}$ and each $k \in \mathbb{N}^+$ it holds (i) $|\Sigma^{p^k} \cap w| \leq 1$ and (ii) if w is defined for all words of length p^k, then $|\Sigma^{p^k} \cap w| = 1$.
(meaning: if $t(i,j) = -p$, then ensure $A_p, B_p \in \mathrm{UP}$, $A_p = \overline{B_p}$, and thus $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ relative to the final oracle.)

V4 For all $i \in \mathbb{N}^+$ with $t(i) = 0$ there exists x such that $F_i^w(x)$ is definite and $\forall_{v \sqsupseteq w} F_i^w(x) \notin \mathcal{CAN}^v$.
(meaning: for every extension of the oracle, F_i is not a proof system for $\mathcal{CAN}$.)

V5 For all $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t(i) \leq c(i,x,F_i^w(x)) < |w|$ it holds $c(i,x,F_i^w(x)) \in w$.
(meaning: if $t(i) > 0$, then from $t(i)$ on, we encode the values of F_i into the oracle. Note that V5 is not in contradiction to V3 or V7 as $|c(\cdot,\cdot,\cdot)|$ is even.)

V6 For all $i \in \mathbb{N}^+$ with $t(i,i) = 0$ there exists x such that $M_i^w(x)$ is definite and has two accepting paths.
(meaning: for every extension of the oracle, M_i is not a UP-machine.)

V7 For all $i \in \mathbb{N}^+$ with $t(i,i) = -q \in \mathbb{P}_{\equiv 1}$ and each $k \in \mathbb{N}^+$ it holds $|\Sigma^{q^k} \cap w| \leq 1$.
(meaning: if $t(i,i) = -q$, ensure that C_q is in UP.)

The subsequent claim follows directly from the definition of t-valid.

Claim 3.3.5 *Let $t, t' \in \mathcal{T}$ such that t' is an extension of t. For all oracles $w \in \Sigma^*$, if w is t'-valid, then w is t-valid.*

Claim 3.3.6 *Let $t \in \mathcal{T}$ and $u, v, w \in \Sigma^*$ such that $u \sqsubseteq v \sqsubseteq w$ and both u and w are t-valid. Then v is t-valid.*

Proof The oracle v satisfies V2, V4, and V6, since u satisfies these conditions. Moreover, v satisfies V3 and V7 as w satisfies these conditions.

Let $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ such that $c(i,x,y) \in v$. Then $c(i,x,y) \in w$ and as w is t-valid, we obtain by V1 that $F_i^w(x) = y \in \mathcal{CAN}^w$. Claim 3.3.4 yields that $F_i^v(x)$ is definite and $F_i^v(x) \in \mathcal{CAN}^v \Leftrightarrow F_i^v(x) \in \mathcal{CAN}^w$. This yields $F_i^v(x) = F_i^w(x) = y$ and $\mathcal{CAN}^v(y) = \mathcal{CAN}^w(y) = 1$. Thus v satisfies V1.

Now let $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ such that $0 < t(i) \leq c(i,x,F_i^v(x)) < |v|$. Again, by Claim 3.3.4, $F_i^v(x)$ is definite and thus $F_i^v(x) = F_i^w(x)$. As $|v| \leq |w|$ and w is t-valid, we obtain by V5 that $c(i,x,F_i^v(x)) = c(i,x,F_i^w(x)) \in w$. Since $v \sqsubseteq w$ and $|v| > c(i,x,F_i^v(x))$, we obtain $c(i,x,F_i^v(x)) \in v$, which shows that v satisfies V5. □

Oracle Construction Let T be an injective enumeration of $\bigcup_{i=1}^{3}(\mathbb{N}^+)^i$ having the property that for all $i,j,r \in \mathbb{N}^+$ the pair (i,j) appears earlier than the triple (i,j,r). Each element of $\bigcup_{i=1}^{3}(\mathbb{N}^+)^i$ is considered as a task. We treat the tasks in the order specified by T and after treating a task we remove it and possibly other tasks from T and continue with the next task that still is in T (i.e., the first task of the current T).

We start with the unique nowhere defined function $t_0 \in \mathcal{T}$ and the unique $w_0 \in \Sigma^*$ with $|w_0| = 1$ and —when considering w_0 as a set— with $w_0 = \emptyset$ (i.e., $w_0 = 0$ when we consider w_0 as a word in Σ^* and $w_0 = 1$ when we consider w_0 as a natural number; cf. the paragraph "Identification of Σ^* and $\mathbb{N}$" on page 31). Observe that w_0 is t_0-valid.

Then we begin treating the tasks. For each task we choose an extension of the current function from $\mathcal{T}$ and a (strict) extension of the oracle. This way we define functions $t_1, t_2, \ldots$ in $\mathcal{T}$ such that t_{i+1} is an extension of t_i and partial oracles $w_0 \sqsubsetneq w_1 \sqsubsetneq w_2 \sqsubsetneq \ldots$ such that each w_i is t_i-valid. So for each task we strictly extend the oracle and are allowed to add more requirements (by extending the respective t_i) that have to be maintained in the further construction.

Finally, we choose $O = \bigcup_{i=0}^{\infty} w_i$. Note that O is totally defined, since in each step we strictly extend the oracle.

We now describe step $s > 0$, which starts with some $t_{s-1} \in \mathcal{T}$ and a t_{s-1}-valid oracle w_{s-1} and treats the first task that still is in T choosing an extension $t_s \in \mathcal{T}$ of t_{s-1} and a t_s-valid $w_s \supsetneq w_{s-1}$ (it will be argued later that the construction we describe below is indeed possible). Let us recall that each task is immediately deleted from T after it is treated. We study five cases depending on the task that is treated in step s.

- task i: Let $t' = t_{s-1} \cup \{i \mapsto 0\}$. If there exists a t'-valid partial oracle $v \supsetneq w_{s-1}$, then let $t_s = t'$ and w_s be the least t'-valid partial oracle $\supsetneq w_{s-1}$. Otherwise, let $t_s = t_{s-1} \cup \{i \mapsto |w_{s-1}|\}$ (note that since $|w_0| = 1$ it holds $t_s(i) = |w_{s-1}| > 0$[4] and that the sufficiently large choice of $t_s(i)$ implies that w_{s-1} is t_s-valid) and choose $w_s = w_{s-1}b$ with $b \in \{0,1\}$ such that w_s is t_s-valid.
  (meaning: try to ensure that F_i is not a proof system for $\mathcal{CAN}$. If this is impossible, require that from now on the values of F_i are encoded into the oracle.)

- task (i,j) with $i \neq j$: Let $t' = t_{s-1} \cup \{(i,j) \mapsto 0\}$. If there exists a t'-valid partial oracle $v \supsetneq w_{s-1}$, then let $t_s = t'$, define w_s to be the least t'-valid partial oracle $\supsetneq w_{s-1}$, and delete all tasks $(i,j,\cdot)$ from T. Otherwise, let $z = |w_{s-1}|$, choose some $p \in \mathbb{P}_{\equiv 3}$ greater than $|z|$ and all $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_{s-1})$, let $t_s = t_{s-1} \cup \{(i,j) \mapsto -p\}$ (note that by the sufficiently large choice of p, the oracle w_{s-1} is t_s-valid), and choose $w_s = w_{s-1}b$ with $b \in \{0,1\}$ such that w_s is t_s-valid.
  (meaning: try to ensure that $(L(M_i), L(M_j))$ is not a disjoint NP-pair. If this is impossible, then choose a sufficiently large prime p and require that the further construction ensures $A_p, B_p \in \mathrm{UP}$, $A_p = \overline{B_p}$, and thus $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ (cf. V3). The treatment of tasks of the form $(i,j,\cdot)$ will make sure that A_p cannot be reduced to $(L(M_i), L(M_j))$.)

- task (i,j,r) with $i \neq j$: It holds $t_{s-1}(i,j) = -p$ for a prime $p \in \mathbb{P}_{\equiv 3}$, since otherwise, this task would have been deleted in the treatment of task (i,j). Define $t_s = t_{s-1}$ and choose a t_s-valid $w_s \supsetneq w_{s-1}$ such that for some $n \in \mathbb{N}^+$ at least one of the following two statements holds:

  - $\forall_{v \supseteq w_s}\, 0^n \in A_p^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
  - $\forall_{v \supseteq w_s}\, 0^n \in B_p^v$, $F_r^{w_s}(0^n)$ is definite, and $M_j^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

[4] Indeed, this is the reason why we do not start with $w_0 = \varepsilon$.

(meaning: make sure that F_r does not $\leq_{\mathrm{m}}^{\mathrm{pp}}$-reduce (A_p, B_p) to $(L(M_i), L(M_j))$. As V3 ensures $A_p = \overline{B_p}$ relative to the final oracle, F_r does not reduce A_p to $(L(M_i), L(M_j))$. Also recall that by V3, it will hold $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ relative to the final oracle.)

- task (i,i): Let $t' = t_{s-1} \cup \{(i,i) \mapsto 0\}$. If there exists a t'-valid partial oracle $v \supsetneq w_{s-1}$, then let $t_s = t'$, define w_s to be the least t'-valid partial oracle $\supsetneq w_{s-1}$, and delete all tasks $(i,i,\cdot)$ from T. Otherwise, let $z = |w_{s-1}|$, choose some $q \in \mathbb{P}_{\equiv 1}$ greater than both $|z|$ and all $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_{s-1})$, let $t_s = t_{s-1} \cup \{(i,i) \mapsto -q\}$ (note that by the sufficiently large choice of q, the oracle w_{s-1} is t_s-valid), and choose $w_s = w_{s-1}b$ with $b \in \{0,1\}$ such that w_s is t_s-valid.
  (meaning: try to ensure that M_i is not a UP-machine. If this is impossible, choose a sufficiently large prime $q \in \mathbb{P}_{\equiv 1}$ and require that in the further construction C_q remains a UP-set (cf. V7). The treatment of tasks of the form $(i,i,\cdot)$ will make sure that C_q cannot be reduced to $L(M_i)$.)

- task (i,i,r): It holds $t_{s-1}(i,i) = -q$ for a prime $q \in \mathbb{P}_{\equiv 1}$, since otherwise, this task would have been deleted in the treatment of task (i,i). Define $t_s = t_{s-1}$ and choose a t_s-valid $w_s \supsetneq w_{s-1}$ such that for some $n \in \mathbb{N}^+$ at least one of the following two conditions holds:

  - $\forall_{v \sqsupseteq w_s}\ 0^n \in C_q^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
  - $\forall_{v \sqsupseteq w_s}\ 0^n \notin C_q^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely accepts.

  (meaning: ensure that F_r does not reduce C_q to $L(M_i)$. Recall that by V7, it will hold $C_q \in \mathrm{UP}$ relative to the final oracle.)

Observe that the choice of t_s guarantees $t_s \in \mathcal{T}$. We now show that the construction is possible. For that purpose, we first describe how a valid oracle can be extended by one bit such that it remains valid.

Claim 3.3.7 *Let $s \in \mathbb{N}$, $w \in \Sigma^*$ be t_s-valid with $w \sqsupseteq w_s$, and $z = |w|$. Then there exists $b \in \{0,1\}$ such that wb is t_s-valid. In detail, the following statements hold.*

1. *If $|z|$ is odd and $|z| \neq p^k$ for all $p \in \mathbb{P}_{\geq 3}$ with $-p \in \mathrm{ran}(t_s)$ and all $k \in \mathbb{N}^+$, then $w0$ and $w1$ are t_s-valid.*

2. *If there exist $p \in \mathbb{P}_{\equiv 3}$ with $-p \in \mathrm{ran}(t_s)$ and $k \in \mathbb{N}^+$ such that $|z| = p^k$, $z \neq 1^{p^k}$, and $w \cap \Sigma^{p^k} = \emptyset$, then $w0$ and $w1$ are t_s-valid.*

3. *If there exist $p \in \mathbb{P}_{\equiv 3}$ with $-p \in \mathrm{ran}(t_s)$ and $k \in \mathbb{N}^+$ such that $z = 1^{p^k}$ and $w \cap \Sigma^{p^k} = \emptyset$, then $w1$ is t_s-valid.*

4. *If there exist $q \in \mathbb{P}_{\equiv 1}$ with $-q \in \mathrm{ran}(t_s)$ and $k \in \mathbb{N}^+$ such that $|z| = q^k$ and $w \cap \Sigma^{q^k} = \emptyset$, then $w0$ and $w1$ are t_s-valid.*

5. *If $z = c(i,x,y)$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$, $0 < t_s(i) \leq z$, and $F_i^w(x) = y$, then $w1$ is t_s-valid.*

6. *If $z = c(i,x,y)$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$, $F_i^w(x) = y \in \mathcal{CAN}^w$, and at least one of the three conditions (i) $t_s(i)$ undefined, (ii) $t_s(i) = 0$, and (iii) $t_s(i) > z$ holds, then $w0$ and $w1$ are t_s-valid.*

7. *In all other cases (i.e., none of the assumptions in 1–6 holds) $w0$ is t_s-valid.*

Proof We first show the following assertions.

$w0$ satisfies V1. (3.9)

If (i) $z = c(i,x,y)$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $F_i^w(x) = y \in \mathcal{CAN}^w$ or (ii) z has odd length, then $w1$ satisfies V1. (3.10)

$w0$ satisfies V5 unless there exist $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ such that (i) $z = c(i,x,y)$, (ii) $0 < t_s(i)$, (iii) $t_s(i) \leq z$, and (iv) $F_i^w(x) = y$. (3.11)

$w1$ satisfies V5. (3.12)

(3.9) and (3.10): Let $i' \in \mathbb{N}^+$ and $x', y' \in \mathbb{N}$ such that $c(i',x',y') \in w$. Then, as w is t_s-valid, by V1, $F_{i'}^w(x') = y' \in \mathcal{CAN}^w$ and by Claim 3.3.4, $F_{i'}^w(x')$ is definite and $y' \in \mathcal{CAN}^v$ for all $v \sqsupseteq w$. Hence in particular, $F_{i'}^{wb}(x') = y' \in \mathcal{CAN}^{wb}$ for all $b \in \{0,1\}$. This shows (3.9). For the proof of (3.10) it remains to consider z. In case (ii) $w1$ satisfies V1 as $|z|$ is odd and each $c(\cdot,\cdot,\cdot)$ has even length. Consider case (i), i.e., $z = c(i,x,y)$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $F_i^w(x) = y \in \mathcal{CAN}^w$. Then by Claim 3.3.4, $F_i^{w1}(x) = y \in \mathcal{CAN}^{w1}$, which shows (3.10).

(3.11) and (3.12): Let $i' \in \mathbb{N}^+$ and $x' \in \mathbb{N}$ such that $0 < t_s(i') \leq c(i',x',F_{i'}^w(x')) < |w|$. Then by Claim 3.3.4, $F_{i'}^w(x')$ is definite and thus $F_{i'}^{wb}(x') = F_{i'}^w(x')$ for all $b \in \{0,1\}$. As w is t_s-valid, it holds $c(i',x',F_{i'}^w(x')) \in w$ and hence $c(i',x',F_{i'}^{wb}(x')) \in w \subseteq wb$ for all $b \in \{0,1\}$. This shows (3.12). In order to finish the proof for (3.11), it remains to consider z. Assume $z = c(i,x,y)$ for some $i,x,y \in \mathbb{N}$ with $i > 0$ (otherwise, $w0$ clearly satisfies V5). If (ii) or (iii) is wrong, then $w0$ satisfies V5. If (iv) is wrong, then $F_i^w(x) \neq y$. By Claim 3.3.4, this computation is definite and hence $F_i^{w0}(x) \neq y$, which is the reason why $w0$ satisfies V5. This shows (3.11).

Let us now prove the assertions 1–7 and note that we do not have to consider V2, V4, and V6 as these conditions are not affected by extending a t_s-valid oracle.

1. By (3.9) and (3.10), the oracles $w0$ and $w1$ satisfy V1. By (3.11) and (3.12), the oracles $w0$ and $w1$ satisfy V5 (for the application of (3.11) recall that each $c(i,x,y)$ has even length and hence for all i,x,y condition (i) does not hold). V3 and V7 are not affected as $|z| \neq p^k$ for all primes p with $-p \in \mathrm{ran}(t_s)$ and all $k > 0$.

2. By (3.9), (3.10), (3.11), and (3.12), the oracles $w0$ and $w1$ satisfy V1 and V5. As $p \in \mathbb{P}_{\equiv 3}$, V7 is satisfied by $w0$ and $w1$. Moreover, $w0$ satisfies V3 as due to $z \neq 1^{p^k}$ the oracle $w0$ is not defined for all words of length p^k. Finally, $w1$ satisfies V3 since $\Sigma^{p^k} \cap w = \emptyset$.

3. By (3.10) and (3.12), the oracle $w1$ satisfies V1 and V5. As $p \in \mathbb{P}_{\equiv 3}$, V7 is satisfied by $w1$. Moreover, as $w \cap \Sigma^{p^k} = \emptyset$, it holds $|w1 \cap \Sigma^{p^k}| = 1$ and hence $w1$ satisfies V3.

4. By (3.9), (3.10), (3.11), and (3.12), the oracles $w0$ and $w1$ satisfy V1 and V5. As $q \in \mathbb{P}_{\equiv 1}$, the oracles $w0$ and $w1$ satisfy V3. Finally, $w0$ trivially satisfies V7 and $w1$ satisfies V7 as $w \cap \Sigma^{q^k} = \emptyset$.

5. By (3.12), the oracle $w1$ satisfies V5. As $|z|$ is even, $w1$ trivially satisfies V3 and V7. It remains to argue for V1. For that purpose we show $y \in \mathcal{CAN}^w$. Then (3.10) can be applied and $w1$ satisfies V1.
   By Claim 3.3.4, $F_i^w(x)$ is definite. Assume that for $z = |w|$ it holds $z = c(i,x,y)$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$, $0 < t_s(i) \leq z$, and $F_i^w(x) = y \notin \mathcal{CAN}^w$. Let $s' > 0$ be the step which treats the task i (note $s' \leq s$ as $t_s(i)$ is defined). By Claim 3.3.5, w is $t_{s'-1}$-valid. Moreover, by Claim 3.3.4, $F_i^w(x) \notin \mathcal{CAN}^v$ for all $v \sqsupseteq w$. Thus w is t'-valid for $t' = t_{s'-1} \cup \{i \mapsto 0\}$, which is why the construction would have chosen $t_{s'} = t'$, in contradiction to $t_s(i) > 0$. Hence $y \in \mathcal{CAN}^w$.

6. By (3.9), (3.10), (3.11), and (3.12), the oracles $w0$ and $w1$ satisfy V1 and V5. As $|z|$ is even, both $w0$ and $w1$ satisfy V3 and V7.

7. By (3.9), $w0$ satisfies V1. Moreover, (3.11) can be applied, since otherwise, there would exist $i, x, y \in \mathbb{N}$ with $i > 0$ such that conditions (i)–(iv) of the assertion (3.11) hold and then we were in case 5. Hence $w0$ satisfies V5. Trivially, $w0$ satisfies V7 and finally, $w0$ satisfies V3 as the only way $w0$ could hurt V3 is that $z = 1^{p^k}$ for some $p \in \mathbb{P}_{\equiv 3}$ with $-p \in \mathrm{ran}(t_s)$ and $k > 0$ as well as $w \cap \Sigma^{p^k} = \emptyset$, but this case is treated in 3.

This finishes the proof of Claim 3.3.7. □

In order to show that the above construction is possible, assume that it is not possible and let $s > 0$ be the least number such that step s of the construction fails. We prove that this assumption leads to a contradiction.

If step s treats a task $\tau \in \mathbb{N}^+ \cup (\mathbb{N}^+)^2$, then $t_{s-1}(\tau)$ is not defined, since the value of τ is defined in the unique treatment of the task τ. If $t_s(\tau)$ is chosen to be 0, then the construction clearly is possible. Otherwise, as was mentioned in the description of the construction, the choice of $t_s(\tau)$ guarantees that the t_{s-1}-valid oracle w_{s-1} is even t_s-valid. Then Claim 3.3.7 ensures that there exists a t_s-valid $w_{s-1}b$ for some $b \in \{0, 1\}$. Hence the construction does not fail in step s, a contradiction.

For the remainder of the proof that the construction above is possible we assume that step s treats a task $(i, j, r) \in (\mathbb{N}^+)^3$. We treat the cases $i = j$ and $i \neq j$ simultaneously whenever it is possible. Recall that in the case $i = j$ we work for the diagonalization ensuring that $L(M_i)$ is not a complete UP-set and in the case $i \neq j$ we work for the diagonalization ensuring that the pair $(L(M_i), L(M_j))$ is not hard for UP ∩ coUP (note: as the task (i, j, r) has not been deleted during the construction, $(L(M_i), L(M_j))$ is a disjoint NP-pair relative to all t_{s-1}-valid extensions of w_{s-1}).

In both cases, $t_s = t_{s-1}$ and $t_s(i, j) = -p$ for some $p \in \mathbb{P}_{\geq 3}$ (recall $p \in \mathbb{P}_{\equiv 1}$ if $i = j$ and $p \in \mathbb{P}_{\equiv 3}$ if $i \neq j$). Let $\gamma(x) = (x^r + r)^{i+j} + i + j$ and choose $n = p^k$ for some $k \in \mathbb{N}^+$ such that

$$2^{2n-2} > 2^{n+1} \cdot \gamma(n) \tag{3.13}$$

and w_{s-1} is undefined for all words of length $\geq n$. Note that by the choice of γ, for each oracle D, all queries of the computations $F_r^D(0^n)$, $M_i^D(F_r^D(0^n))$, and $M_j^D(F_r^D(0^n))$ are of length $\leq \gamma(n)$.

We define $u \sqsupseteq w_{s-1}$ to be the minimal t_s-valid partial oracle that is defined for all words of length $< n$. Such an oracle exists by Claim 3.3.7.

Moreover, for $z \in \Sigma^n$, let $u_z \sqsupsetneq u$ be the minimal t_s-valid partial oracle with $u_z \cap \Sigma^n = \{z\}$ that is defined for all words of length $\leq \gamma(n)$. Such an oracle exists by Claim 3.3.7: first, starting from u we extend the current oracle bitwise such that (i) it remains t_s-valid, (ii) it is defined for precisely the words of length $\leq n$, and (iii) its intersection with Σ^n equals $\{z\}$. This is possible by (2, 3, and 7) or (4 and 7) of Claim 3.3.7 depending on whether $p \in \mathbb{P}_{\equiv 3}$ or $p \in \mathbb{P}_{\equiv 1}$. Then by Claim 3.3.7, the current oracle can be extended bitwise without losing its t_s-validity until it is defined for all words of length $\leq \gamma(n)$.

Claim 3.3.8 *Let $z \in \Sigma^n$.*

1. *For each $\alpha \in u_z \cap \Sigma^{>n}$ one of the following statements holds.*

   - *$\alpha = 1^{p'^\kappa}$ for some $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and some $\kappa > 0$.*
   - *$\alpha = c(i', x, y)$ for some $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq c(i', x, y)$, $F_{i'}^{u_z}(x) = y$, and $y \in \mathcal{CAN}^{u_z}$.*

2. *For all* $p' \in \mathbb{P}_{\equiv 3}$ *with* $-p' \in \mathrm{ran}(t_s)$ *and all* $\kappa > 0$, *if* $n < p'^{\kappa} \leq \gamma(n)$, *then* $u_z \cap \Sigma^{p'^{\kappa}} = \{1^{p'^{\kappa}}\}$.

Proof

1. Let $\alpha \in u_z \cap \Sigma^{>n}$. Moreover, let u' be the prefix of u_z that has length α, i.e., α is the least word that u' is not defined for. In particular, it holds $u' \cap \Sigma^{\leq n} = u_z \cap \Sigma^{\leq n}$ and thus $u' \cap \Sigma^n = \{z\}$. As $u \sqsubseteq u' \sqsubseteq u_z$ and both u and u_z are t_s-valid, Claim 3.3.6 yields that u' is also t_s-valid.
   Let us apply Claim 3.3.7 to the oracle u'. If one of the cases 1, 2, 4, 6, and 7 can be applied, then $u'0$ is t_s-valid and can be extended to a t_s-valid oracle u'' with $|u''| = |u_z|$ by Claim 3.3.7. As u'' and u_z agree on all words $< \alpha$ and $\alpha \in u_z - u''$, we obtain $u'' < u_z$ and due to $u' \sqsubseteq u''$ we know that $u'' \cap \Sigma^n = \{z\}$. This is a contradiction to the choice of u_z (recall that u_z is the minimal t_s-valid partial oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies $u_z \cap \Sigma^n = \{z\}$).
   Hence none of the cases 1, 2, 4, 6, and 7 of Claim 3.3.7 can be applied, i.e., either (i) $\alpha = 1^{p'^{\kappa}}$ for some $p' \in \mathbb{P}_{\equiv 3}$ and $\kappa > 0$ with $-p' \in \mathrm{ran}(t_s)$ or (ii) $\alpha = c(i', x, y)$ for $i', x, y \in \mathbb{N}$, $i' > 0$, and $0 < t_s(i') \leq \alpha$. In the latter case, as $\alpha \in u_z$ and u_z is t_s-valid, we obtain from V1 that $F_{i'}^{u_z}(x) = y \in \mathcal{CAN}^{u_z}$.

2. As $-p' \in \mathrm{ran}(t_s)$, u_z is t_s-valid, and u_z is defined for all words of length p'^{κ}, V3 yields that there exists $\beta \in \Sigma^{p'^{\kappa}} \cap u_z$. Let β be the minimal element of $\Sigma^{p'^{\kappa}} \cap u_z$. It suffices to show $\beta = 1^{p'^{\kappa}}$. For a contradiction, we assume $\beta < 1^{p'^{\kappa}}$. Let u' be the prefix of u_z that is defined for exactly the words of length $< p'^{\kappa}$. Then $u \sqsubseteq u' \sqsubseteq u_z$ and both u and u_z are t_s-valid. Hence by Claim 3.3.6, the oracle u' is t_s-valid as well.
   By Claim 3.3.7, u' can be extended to a t_s-valid oracle u'' that satisfies $|u''| = |u_z|$ and $u'' \cap \Sigma^{p'^{\kappa}} = \{1^{p'^{\kappa}}\}$. Then $\beta \in u_z - u''$. As the oracles u'' and u_z agree on all words smaller than β, we have $u'' < u_z$ and $u'' \cap \Sigma^n = \{z\}$, in contradiction to the choice of u_z (again, recall that u_z is the minimal t_s-valid partial oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies $u_z \cap \Sigma^n = \{z\}$).

This finishes the proof of Claim 3.3.8. □

Let us study the case that for some odd (resp., even) $z \in \Sigma^n$ the computation $M_i^{u_z}(F_r^{u_z}(0^n))$ (resp., $M_j^{u_z}(F_r^{u_z}(0^n))$ if z is even) rejects. Then since u_z is defined for all words of length $\leq \gamma(n)$, the aforementioned computation even definitely rejects and the computation $F_r^{u_z}(0^n)$ is definite. If $i \neq j$, then $p \in \mathbb{P}_{\equiv 3}$ and since $z \in u_z$, we have $0^n \in A_p^v$ for all $v \sqsupseteq u_z$ (resp., $0^n \in B_p^v$ for all $v \sqsupseteq u_z$ if z is even). Analogously, if $i = j$, then $p \in \mathbb{P}_{\equiv 1}$ and as $z \in u_z$, we have $0^n \in C_p^v$ for all $v \sqsupseteq u_z$. Hence in all these cases we can choose $w_s = u_z$ and obtain a contradiction to the assumption that step s of the construction fails in treating the task (i, j, r). Therefore, for the remainder of the proof that the construction is possible we assume the following:

- For each odd $z \in \Sigma^n$ the computation $M_i^{u_z}(F_r^{u_z}(0^n))$ definitely accepts.
- For each even $z \in \Sigma^n$ the computation $M_j^{u_z}(F_r^{u_z}(0^n))$ definitely accepts.

Note that in case $i = j$ we could have also formulated the two conditions equivalently in the following simpler way: for each $z \in \Sigma^n$ the computation $M_i^{u_z}(F_r^{u_z}(0^n))$ definitely accepts. Recall, however, that as far as possible we consider the cases $i = j$ and $i \neq j$ simultaneously.

We will show that this assumption leads to a contradiction, which will ensure that the above construction is possible.

Let U_z for $z \in \Sigma^n$ odd (resp., $z \in \Sigma^n$ even) be the set that consists of all oracle queries of the computation $F_r^{u_z}(0^n)$ and all oracle queries of the least accepting path of $M_i^{u_z}(F_r^{u_z}(0^n))$ (resp., $M_j^{u_z}(F_r^{u_z}(0^n))$). Observe $\ell(U_z) \leq \gamma(n)$. Moreover, define $Q_0(U_z) = U_z$ and for $m \in \mathbb{N}$,

$$Q_{m+1}(U_z) = \bigcup_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q_m(U_z)}} \Big[\{q \mid q \text{ is queried by } F_{i'}^{u_z}(x)\}\cup \{q \mid y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x'\rangle \text{ for some } i'' > 0 \text{ and } x' \in \mathbb{N},\ M_{i''}^{u_z}(x') \text{ has an accepting path, and } q \text{ is queried by the least such path}\}\Big].$$

Let $Q(U_z) = \bigcup_{m\in\mathbb{N}} Q_m(U_z)$.

Claim 3.3.9 *For all $z \in \Sigma^n$, $\ell(Q(U_z)) \leq 2\ell(U_z) \leq 2\gamma(n)$ and the length of each word in $Q(U_z)$ is $\leq \gamma(n)$.*

Proof We show that for all $m \in \mathbb{N}$, $\ell(Q_{m+1}(U_z)) \leq 1/2 \cdot \ell(Q_m(U_z))$. Then $\sum_{m=0}^{\kappa} 1/2^m \leq 2$ for all $\kappa \in \mathbb{N}$ implies $\ell(Q(U_z)) \leq 2 \cdot \ell(U_z)$. Moreover, the second part of the claim follows from $\ell(U_z) \leq \gamma(n)$ and $\ell(Q_{m+1}(U_z)) \leq 1/2 \cdot \ell(Q_m(U_z))$.

Let $m \in \mathbb{N}$ and consider an arbitrary element α of $Q_m(U)$. If α is not of the form $c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$, then α generates no elements in $Q_{m+1}(U)$. Assume $\alpha = c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$. The computation $F_{i'}^v(x)$ runs for at most $|x|^{i'} + i' < |\alpha|/4$ steps, where "$<$" holds by (3.8). Hence the set of queries Q of $F_{i'}^v(x)$ satisfies $\ell(Q) \leq |\alpha|/4$.

Let us assume $y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x'\rangle$ for $i'' \in \mathbb{N}^+$ and $x' \in \mathbb{N}$ (otherwise, the second of the two sets in the definition of $Q_{m+1}(U_z)$ is empty and does not require further consideration). Then the computation $M_{i''}^v(x)$ runs for less than $|x'|^{i''} + i'' < |y| < |\alpha|/4$ steps, where again "$<$" holds by (3.8). Hence for the set Q of queries of the least accepting path of the computation $M_{i''}^v(x)$ (if such a path exists) we have $\ell(Q) \leq |\alpha|/4$. Consequently,

$$\begin{aligned}
\ell(Q_{m+1}(U)) &\leq \sum_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q_m(U_z)}} \Big[\underbrace{\ell\big(\{q \mid q \text{ is queried by } F_{i'}^{u_z}(x)\}\big)}_{\leq |c(i',x,y)|/4} + \\
&\qquad \underbrace{\ell\big(\{q \mid y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x'\rangle \text{ for some } i'' > 0 \text{ and } x' \in \mathbb{N}, M_{i''}^{u_z}(x') \text{ has an accepting path, and } q \text{ is queried by the least such path}\}\big)\Big]}_{\leq |c(i',x,y)|/4} \\
&\leq \sum_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q_m(U_z)}} |c(i',x,y)|/2 \\
&\leq \ell(Q_m(U_z))/2,
\end{aligned}$$

which finishes the proof of Claim 3.3.9. □

We now define a similar notion. Let $Q'_0(U_z) = U_z$ and for $m \in \mathbb{N}$,

$$Q'_{m+1}(U_z) = \bigcup_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q'_m(U_z)}} \{q \mid q \text{ is queried by } F_{i'}^{u_z}(x)\}.$$

Moreover, define $Q'(U_z) = \bigcup_{m\in\mathbb{N}} Q'_m(U_z)$. By definition $Q'_m(U_z) \subseteq Q_m(U_z)$ for all $m \in \mathbb{N}$ and hence $Q'(U_z) \subseteq Q(U_z) \subseteq \Sigma^{\leq\gamma(n)}$, where the latter "$\subseteq$" holds by Claim 3.3.9.

For $z, z' \in \Sigma^n$ we say that z and z' *conflict* (resp., *strongly conflict*) if there is a word $\alpha \in Q(U_z) \cap Q(U_{z'})$ (resp., $\alpha \in Q'(U_z) \cap Q'(U_{z'})$) which is in $u_z \triangle u_{z'}$. In that case, we say z and z' (*strongly*) *conflict in* α. Note that whenever z and z' (strongly) conflict in a word α, then $|\alpha| \geq n$, as u_z and $u_{z'}$ are both extensions of u and u is defined for all words of length $< n$. By definition, if z and z' strongly conflict, then z and z' conflict.

The next four claims are dedicated to the purpose of proving that for each odd $z \in \Sigma^n$ and each even $z' \in \Sigma^n$, it holds that z and z' conflict in a word of length n. Indeed, then z and z' conflict in one of the words z and z' as these are the only words of length n in $u_z \cup u_{z'}$. Once we have proven this, we will be able to generate a contradiction that completes the proof that the oracle construction described above is possible.

Claim 3.3.10 *Let $z, z' \in \Sigma^n$ such that z is odd and z' is even. If z and z' conflict, then they conflict in a word of length n.*

Proof Let α be the least word in which z and z' conflict (note that $|\alpha| \leq \gamma(n)$ due to $\alpha \in Q(U_z) \cap Q(U_{z'})$ and Claim 3.3.9). Then $\alpha \in u_z \triangle u_{z'}$. By symmetry, it suffices to consider the case $\alpha \in u_z - u_{z'}$. For a contradiction, assume that $|\alpha| \neq n$, which implies $|\alpha| > n$ as $u_z \sqsupseteq u$, $u_{z'} \sqsupseteq u$, and u is defined for all words of length $< n$. Then by Claim 3.3.8, two situations are possible.

1. If $\alpha = 1^{p'^\kappa}$ for $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$, then by Claim 3.3.8.2, $\alpha \in u_{z'}$, a contradiction.

2. Here $\alpha = c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq c(i', x, y)$ and $F_{i'}^{u_z}(x) = y \in \mathcal{CAN}^{u_z}$. Then $F_{i'}^{u_{z'}}(x) \neq y$, since otherwise, by the t_s-validity of $u_{z'}$ and V5, it would hold $\alpha \in u_{z'}$. Consequently, $F_{i'}^{u_{z'}}(x) \neq F_{i'}^{u_z}(x)$. Hence there exists a query β that is asked by both $F_{i'}^{u_z}(x)$ and $F_{i'}^{u_{z'}}(x)$ and that is in $u_z \triangle u_{z'}$ (otherwise, both computations would output the same word). By definition of $Q(U_z)$ and $Q(U_{z'})$, it holds $\beta \in Q(U_z) \cap Q(U_{z'})$. Hence z and z' conflict in β and $|\beta| \leq |x|^{i'} + i' < |c(i', x, y)| = |\alpha|$, in contradiction to the assumption that α is the least word which z and z' conflict in.

In both cases we obtain a contradiction. Thus the proof is complete. □

We want to show next that for all odd $z \in \Sigma^n$ and all even $z' \in \Sigma^n$ it holds that z and z' indeed conflict.

Recall that $u \sqsupseteq w_{s-1}$ is the minimal t_s-valid partial oracle that is defined for all words of length $< n$. Let $z \in \Sigma^n$. We say that a partial oracle $v \sqsupseteq u$ is *consistent* with u_z relative to $Q(U_z)$ (resp., relative to $Q'(U_z)$) if $v(q) = u_z(q)$ for all $q \in Q(U_z)$ (resp., all $q \in Q'(U_z)$) that v is defined for (i.e., $q < |v|$). Note that since v and u_z are both extensions of u, it trivially holds $v(q) = u_z(q)$ for all $q \in \Sigma^{<n}$.

As an abbreviation, whenever we say that v is consistent with u_z, then we mean that v is consistent with u_z *relative to* $Q(U_z)$.

Claim 3.3.11 *The following statements hold.*

1. *Let $t = t_{s'}$ for some $0 \leq s' \leq s$ and $z, z' \in \Sigma^n$ such that z and z' do not conflict. For each t-valid partial oracle $v \sqsupsetneq u$ that is defined for exactly the words of length $\leq n$ and consistent with both u_z and $u_{z'}$, there exists a t-valid partial oracle $v' \sqsupsetneq v$ that is consistent with both u_z and $u_{z'}$ and satisfies $|v'| = |u_z|$.*

2. *Let $z \in \Sigma^n$. For each t_s-valid partial oracle $v \sqsupsetneq u$ that is defined for exactly the words of length $\leq n$ and consistent with u_z relative to $Q'(U_z)$, there exists a t_s-valid partial oracle $v' \sqsupsetneq v$ that is consistent with u_z relative to $Q'(U_z)$ and satisfies $|v'| = |u_z|$.*

Note that in the notation of the claim above, v is consistent with u_z if and only if for all $q \in Q(U_z) \cap \Sigma^n$, it holds $v(q) = 1$ if $q = z$ and $v(q) = 0$ otherwise. Moreover, recall that $|v'| = |u_z|$ expresses that v' is defined for exactly the words of length $\leq \gamma(n)$.

Proof of Claim 3.3.11 The two statements can be proven in a similar way. Basically, the proof of the second statement is a simplified version of the proof of the first statement. Nevertheless, for the sake of completeness, we also give a separate proof for the second statement.

1. Let $w \sqsupseteq v$ with $|w| < |u_z|$ be t-valid and consistent with u_z and $u_{z'}$. Moreover, let $\alpha = |w|$, i.e., α is the least word that w is not defined for. It suffices to show the following:

   (a) If $\alpha = 0^{p'^\kappa}$ for a $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$, then there exists a t-valid $w' \sqsupsetneq w$ that is defined for exactly the words of length $\leq p'^\kappa$ and consistent with u_z and $u_{z'}$.

   Note that in this case $|w'| \leq |u_z|$, since u_z is defined for exactly the words of length $\leq \gamma(n)$.

   (b) If for all $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and all $\kappa > 0$ the word α is not of length p'^κ, then there exists $b \in \{0,1\}$ such that wb is t-valid and consistent with u_z and $u_{z'}$.

   (a) Assume $\alpha = 0^{p'^\kappa}$ for some $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$. Then we let $w' \sqsupsetneq w$ be the minimal partial oracle that is defined for all words of length $\leq p'^\kappa$ and contains $1^{p'^\kappa}$, i.e., $w' = w \cup \{1^{p'^\kappa}\}$ when interpreting the partial oracles as sets. As $u_z \cap \Sigma^{p'^\kappa} = u_{z'} \cap \Sigma^{p'^\kappa} = \{1^{p'^\kappa}\}$ by Claim 3.3.8.2 (note $|\alpha| > n$, since $\alpha = |w|$, $w \sqsupseteq v$, and v is defined for all words of length $\leq n$), we obtain that w' is consistent with u_z and $u_{z'}$. Moreover, if $-p' \in \mathrm{ran}(t)$, then w' is t-valid by Claim 3.3.7.2 and Claim 3.3.7.3. If $-p' \notin \mathrm{ran}(t)$, then w' is t-valid by Claim 3.3.7.1.

   (b) Here we study two subcases.

      i. Assume that $\alpha = c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq \alpha$. Let us first assume that $\alpha \notin Q(U_z) \cup Q(U_{z'})$. Then there exists $b \in \{0,1\}$ such that wb is t-valid (cf. Claim 3.3.7) and clearly wb is consistent with u_z and $u_{z'}$.
      From now on we assume $\alpha \in Q(U_z) \cup Q(U_{z'})$. By symmetry, it suffices to consider the case $\alpha \in Q(U_z)$. As all queries q of $F_{i'}^{u_z}(x)$ are in $Q(U_z)$ and satisfy $u_z(q) = w(q)$ due to $|q| \leq |x|^{i'} + i' < \alpha$, it holds

$$F_{i'}^{w}(x) = F_{i'}^{u_z}(x). \tag{3.14}$$

      We study two cases.

      A. If $\alpha \in u_z$, then by V1, $F_{i'}^{u_z}(x) = y \in \mathcal{CAN}^{u_z}$. By (3.14), $F_{i'}^{w}(x) = y$. Let us show $y \in \mathcal{CAN}^{w}$: As $y \in \mathcal{CAN}^{u_z}$, it holds $y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x' \rangle$ for some $i'' > 0$ and $x' \in \mathbb{N}$, the computation $M_{i''}^{u_z}(x')$ has an accepting path, and all queries q of the least accepting path of this computation are in $Q(U_z)$ and thus satisfy $u_z(q) = w(q)$ (note $|q| \leq |x'|^{i''} + i'' < |y| < |\alpha|$). Hence $M_{i''}^{w}(x')$ accepts and $y \in \mathcal{CAN}^{w}$. Let us choose $b = 1$. Note that $t(i')$ is not necessarily defined. If $t(i')$ is defined, then $0 < t(i') = t_s(i') \leq \alpha$, we apply Claim 3.3.7.5, and obtain that wb is t-valid. If $t(i')$ is undefined, then we apply Claim 3.3.7.6 and obtain that wb is t-valid. Clearly wb is consistent with u_z. In order to see that wb is consistent with $u_{z'}$, it suffices to show $\big(\alpha \in Q(U_{z'}) \Rightarrow \alpha \in u_{z'}\big)$. This holds since otherwise, z and z' conflict.

B. Assume $\alpha \notin u_z$. Then by V5, $F_{i'}^{u_z}(x) \neq y$. By (3.14), $F_{i'}^{w}(x) \neq y$. Choose $b = 0$. Then by Claim 3.3.7.7, wb is t-valid and clearly wb is consistent with u_z. In order to see that wb is consistent with $u_{z'}$, it suffices to argue for α. If $\alpha \in Q(U_{z'})$, then $\alpha \notin u_{z'}$ as otherwise, z and z' would conflict.

ii. We now consider the remaining cases, i.e., we may assume
- α is not of length p'^{κ} for all $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and all $\kappa > 0$ and
- $\alpha \neq c(i', x, y)$ for all $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq \alpha$.

In this case, it holds $\alpha \notin u_z \cup u_{z'}$ by Claim 3.3.8.1. We choose $b = 0$ and obtain that wb is consistent with u_z and $u_{z'}$. Moreover, by Claim 3.3.7, wb is t-valid ($w0$ is t-valid unless Claim 3.3.7.3 or Claim 3.3.7.5 is applicable, which is —by assumption— not the case).

2. Let $w \sqsupseteq v$ with $|w| < |u_z|$ be t_s-valid and consistent with u_z relative to $Q'(U_z)$. Moreover, let $\alpha = |w|$. It suffices to show the following:

(a) If $\alpha = 0^{p'^{\kappa}}$ for a $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$, then there exists a t_s-valid $w' \supsetneq w$ that is defined for exactly the words of length $\leq p'^{\kappa}$ and consistent with u_z relative to $Q'(U_z)$.
Note that in this case $|w'| \leq |u_z|$, since u_z is defined for exactly the words of length $\leq \gamma(n)$.

(b) If for all $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and all $\kappa > 0$ the word α is not of length p'^{κ}, then there exists $b \in \{0, 1\}$ such that wb is t_s-valid and consistent with u_z relative to $Q'(U_z)$.

(a) Assume $\alpha = 0^{p'^{\kappa}}$ for some $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$. Then we let $w' \supsetneq w$ be the minimal partial oracle that is defined for all words of length $\leq p'^{\kappa}$ and contains $1^{p'^{\kappa}}$, i.e., $w' = w \cup \{1^{p'^{\kappa}}\}$ when interpreting the partial oracles as sets. As $u_z \cap \Sigma^{p'^{\kappa}} = \{1^{p'^{\kappa}}\}$ by Claim 3.3.8.2 (note $|\alpha| > n$, since $\alpha = |w|$, $w \sqsupseteq v$, and v is defined for all words of length $\leq n$), we obtain that w' is consistent with u_z relative to $Q'(U_z)$. Moreover, w' is t_s-valid by Claim 3.3.7.2 and Claim 3.3.7.3.

(b) Here we study two subcases.

i. Assume that $\alpha = c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq \alpha$. Let us first assume that $\alpha \notin Q'(U_z)$. Then there exists $b \in \{0, 1\}$ such that wb is t_s-valid (cf. Claim 3.3.7) and clearly wb is consistent with u_z relative to $Q'(U_z)$. From now on we assume $\alpha \in Q'(U_z)$. As all queries q of $F_{i'}^{u_z}(x)$ are in $Q'(U_z)$ and satisfy $u_z(q) = w(q)$ due to $|q| \leq |x|^{i'} + i' < \alpha$, it holds

$$F_{i'}^{w}(x) = F_{i'}^{u_z}(x). \tag{3.15}$$

We study two cases. If $\alpha \in u_z$, then by V1, $F_{i'}^{u_z}(x) = y \in \mathcal{CAN}^{u_z}$. By (3.15), $F_{i'}^{w}(x) = y$. Let us choose $b = 1$. Applying Claim 3.3.7.5, we obtain that wb is t_s-valid. Clearly wb is consistent with u_z relative to $Q'(U_z)$.
Assume $\alpha \notin u_z$. Then by V5, $F_{i'}^{u_z}(x) \neq y$. By (3.15), $F_{i'}^{w}(x) \neq y$. Choose $b = 0$. Then by Claim 3.3.7.7, wb is t_s-valid and clearly wb is consistent with u_z relative to $Q'(U_z)$.

ii. We now consider the remaining cases, i.e., we may assume that
- for all $p' \in \mathbb{P}_{\equiv 3}$ with $-p' \in \mathrm{ran}(t_s)$ and all $\kappa > 0$ the number α is not of length p'^{κ} and

- for all $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i') \leq \alpha$ it holds $\alpha \neq c(i', x, y)$.

In this case, it holds $\alpha \notin u_z$ by Claim 3.3.8.1. We choose $b = 0$ and obtain that wb is consistent with u_z relative to $Q'(U_z)$. Moreover, by Claim 3.3.7, wb is t_s-valid ($wb = w0$ is t_s-valid unless Claim 3.3.7.3 or Claim 3.3.7.5 is applicable, which is —by assumption— not the case).

This finishes the proof of Claim 3.3.11. □

Claim 3.3.12 *For all $z \in \Sigma^n$ it holds $z \in Q'(U_z)$.*

Proof For a contradiction, assume $z \notin Q'(U_z)$ for some $z \in \Sigma^n$. We study the cases $i = j$ and $i \neq j$ separately.

First let us consider the case $i \neq j$. Here $p \in \mathbb{P}_{\equiv 3}$. By symmetry, it suffices to consider the case that z is odd. Let z' be the minimal even element of Σ^n that is not in $Q'(U_z)$. Such a number z' exists as $2^{n-1} > 4\gamma(n) > 2\gamma(n)$ by (3.13), $\ell(Q'(U_z)) \leq \ell(Q(U_z)) \leq 2\gamma(n)$ by Claim 3.3.9, and hence $|Q'(U_z)| \leq \ell(Q'(U_z)) \leq 2\gamma(n) < 2^{n-1} = |\{z'' \in \Sigma^n \mid z'' \text{ even}\}|$. Now choose v to be the partial oracle that is defined for exactly the words of length $\leq n$ and that satisfies $v = u \cup \{z'\}$ when the partial oracles are considered as sets. Then v is t_s-valid by Claim 3.3.7.2, Claim 3.3.7.3, and Claim 3.3.7.7. Moreover, as $z, z' \notin Q'(U_z)$, the oracle v is consistent with u_z relative to $Q'(U_z)$. Hence we can apply Claim 3.3.11.2 to the oracle v for the parameter z and obtain a t_s-valid partial oracle v' that is defined for all words of length $\leq \gamma(n)$, satisfies $v' \cap \Sigma^n = \{z'\}$, and is consistent with u_z relative to $Q'(U_z)$. Hence $v'(q) = u_z(q)$ for all $q \in Q'(U_z)$. By this property and by the fact that $U_z \subseteq Q'(U_z)$ contains all queries asked by $F_r^{u_z}(0^n)$ and all queries asked by the least accepting path of $M_i^{u_z}(F_r^{u_z}(0^n))$ (such a path exists by the assumptions made on page 60), it holds that $F_r^{v'}(0^n) = F_r^{u_z}(0^n)$ and that the least accepting path of $M_i^{u_z}(F_r^{u_z}(0^n))$ is also an accepting path of the computation $M_i^{v'}(F_r^{v'}(0^n))$. As v' is defined for all words of length $\leq \gamma(n)$, the computation $M_i^{v'}(F_r^{v'}(0^n))$ definitely accepts. Let us study two cases depending on whether $M_j^{v'}(F_r^{v'}(0^n))$ definitely accepts or definitely rejects (note that this computation is definite as v' is defined for all words of length $\leq \gamma(n)$):

- Assume that $M_j^{v'}(F_r^{v'}(0^n))$ definitely accepts. Let $s' > 0$ be the step that treats the task (i, j). Hence $s' < s$ since $t_s(i, j)$ is defined. By Claim 3.3.5, the oracle v' is $t_{s'-1}$-valid. Now, as both $M_i^{v'}(F_r^{v'}(0^n))$ and $M_j^{v'}(F_r^{v'}(0^n))$ definitely accept, v' is even t'-valid for $t' = t_{s'-1} \cup \{(i, j) \mapsto 0\}$. But then the construction would have chosen $t_{s'} = t'$, in contradiction to $t_s(i, j) \neq 0$.

- Assume that $M_j^{v'}(F_r^{v'}(0^n))$ definitely rejects. Since v' is defined for all words of length $\leq \gamma(n)$, the computation $F_r^{v'}(0^n)$ is definite. As $v' \cap \Sigma^n = \{z'\}$, it holds $0^n \in B_p^{\widetilde{v}}$ for all $\widetilde{v} \sqsupseteq v'$. This is a contradiction to the assumption that step s of the construction fails.

As in both cases we obtain a contradiction, the proof for the case $i \neq j$ is complete.

Now assume $i = j$. Here, $p \in \mathbb{P}_{\equiv 1}$. Let v be the partial oracle that is defined for exactly the words of length $\leq n$ and satisfies $v = u$ when the partial oracles are considered as sets. Then v is t_s-valid by Claim 3.3.7.4 and v is consistent with u_z relative to $Q'(U_z)$ as both oracles are extensions of u, $u_z \cap \Sigma^n = \{z\}$, and $z \notin Q'(U_z)$. Thus we can apply Claim 3.3.11.2 to the oracle v for the parameter z. Hence there exists a t_s-valid partial oracle v' that satisfies $|v'| = |u_z|$, that satisfies $v \cap \Sigma^n = \emptyset$, and that is consistent with u_z relative to $Q'(U_z)$. The first condition expresses that v' is defined for all words of length $\leq \gamma(n)$, in particular for all words in $Q'(U_z)$). Thus, as v' is consistent with u_z, it holds $v'(q) = u_z(q)$ for all $q \in Q'(U_z)$. By this property and

the fact that $U_z \subseteq Q'(U_z)$ contains all queries of the computation $F_r^{u_z}(0^n)$ and all queries asked by the least accepting path of $M_i^{u_z}(F_r^{u_z}(0^n))$ (such a path exists by the assumptions made on page 60), it holds that $F_r^{u_z}(0^n) = F_r^{v'}(0^n)$ and that the least accepting path of $M_i^{u_z}(F_r^{u_z}(0^n))$ is also an accepting path of the computation $M_i^{v'}(F_r^{v'}(0^n))$. As v' is defined for all words of length $\leq \gamma(n)$, the computations $F_r^{v'}(0^n)$ and $M_i^{v'}(F_r^{v'}(0^n))$ are definite. Putting things together, we have that $0^n \notin C_q^{\widetilde{v}}$ for all $\widetilde{v} \sqsupseteq v'$, $F_r^{v'}(0^n)$ is definite, and $M_i^{v'}(F_r^{v'}(0^n))$ definitely accepts, in contradiction to the assumption that step s of the construction fails. □

Claim 3.3.13 *For all odd $z \in \Sigma^n$ and all even $z' \in \Sigma^n$, it holds that z and z' conflict.*

Proof Assume there are z odd and z' even such that z and z' do not conflict. Then let $v \sqsupsetneq u$ be the minimal partial oracle that is defined for all words of length $\leq n$ and contains z and z', i.e., interpreting partial oracles as sets it holds $v = u \cup \{z, z'\}$. Let $s' > 0$ be the step that treats the task (i, j). Then $s' < s$ as $t_s(i, j)$ is defined. As $t_s \in \mathcal{T}$ is injective on its support and $t_s(i, j) = -p$, it holds $-p \notin \operatorname{ran}(t_{s'-1})$. Therefore, v is $t_{s'-1}$-valid by Claim 3.3.7.1 (recall that u is $t_{s'-1}$-valid by Claim 3.3.5). In order to apply Claim 3.3.11.1 to v for the parameters z, z', and $s' - 1$, it is sufficient to show that v is consistent with u_z (relative to $Q(U_z)$) and v is consistent with $u_{z'}$ (relative to $Q(U_{z'})$). Assume this is not the case. Then due to $u_z \cap \Sigma^n = \{z\}$ and $u_{z'} \cap \Sigma^n = \{z'\}$ it holds $z \in Q(U_{z'})$ or $z' \in Q(U_z)$. As by Claim 3.3.12, $z \in Q'(U_z) \subseteq Q(U_z)$ and $z' \in Q'(U_{z'}) \subseteq Q(U_{z'})$, one of the words z and z' is in $Q(U_z) \cap Q(U_{z'})$. Since u_z and $u_{z'}$ disagree on both z and z', it holds that z and z' conflict, a contradiction. Hence Claim 3.3.11.1 can be applied to v for the parameters z, z', and $s' - 1$.

Applying Claim 3.3.11.1, we obtain a $t_{s'-1}$-valid $v' \sqsupseteq v$ that is defined for all words of length $\leq \gamma(n)$ and is consistent with u_z (relative to $Q(U_z)$) and consistent with $u_{z'}$ (relative to $Q(U_{z'})$), i.e., $v'(q) = u_z(q)$ for all $q \in Q(U_z)$ and $v(q) = u_{z'}(q)$ for all $q \in Q(U_{z'})$ (recall $Q(U_z) \cup Q(U_{z'}) \subseteq \Sigma^{\leq \gamma(n)}$, which is the reason why v' is defined for all words in $Q(U_z) \cup Q(U_{z'})$). We claim

$$v' \text{ is } t'\text{-valid for } t' = t_{s'-1} \cup \{(i, j) \mapsto 0\}. \tag{3.16}$$

Once (3.16) is proven, we obtain a contradiction as then the construction would have chosen $t_{s'} = t'$, in contradiction to $t_s(i, j) \neq 0$. Then our assumption is wrong and for all odd $z \in \Sigma^n$ and all even $z' \in \Sigma^n$, it holds that z and z' conflict.

It remains to prove (3.16). We study two cases.

Case 1: first we assume $i \neq j$. In this case it suffices to prove that $M_i^{v'}(F_r^{v'}(0^n))$ and $M_j^{v'}(F_r^{v'}(0^n))$ definitely accept. Recall that $M_i^{u_z}(F_r^{u_z}(0^n))$ and $M_j^{u_{z'}}(F_r^{u_{z'}}(0^n))$ definitely accept, $v'(q) = u_z(q)$ for all $q \in Q(U_z)$, and $v'(q) = u_{z'}(q)$ for all $q \in Q(U_{z'})$. As all queries of the computations $F_r^{u_z}(0^n)$ and $M_i^{u_z}(F_r^{u_z}(0^n))$ are in $Q(U_z)$ and all queries of the computations $F_r^{u_{z'}}(0^n)$ and $M_j^{u_{z'}}(F_r^{u_{z'}}(0^n))$ are in $Q(U_{z'})$, we obtain that the least accepting paths of $M_i^{u_z}(F_r^{u_z}(0^n))$ and $M_j^{u_{z'}}(F_r^{u_{z'}}(0^n))$ are also accepting paths of the computations $M_i^{v'}(F_r^{v'}(0^n))$ and $M_j^{v'}(F_r^{v'}(0^n))$. Moreover, the latter computations are definite by the choice of v'. Thus v' is t'-valid.

Case 2: assume $i = j$. We have to prove that on some input x the computation $M_i^{v'}(x)$ has at least two accepting paths. By Claim 3.3.12, $z \in Q'(U_z)$ and $z' \in Q'(U_{z'})$. As z and z' do not conflict, z and z' do not strongly conflict and it holds $z \notin Q'(U_{z'})$, which implies $Q'(U_z) \neq Q'(U_{z'})$. Let $\kappa \in \mathbb{N}$ be minimal such that $Q'_\kappa(U_z) \neq Q'_\kappa(U_{z'})$ and for a contradiction, assume $\kappa > 0$.

Let $\alpha \in Q'_\kappa(U_z) \triangle Q'_\kappa(U_{z'})$. Without loss of generality, we assume $\alpha \in Q'_\kappa(U_z) - Q'_\kappa(U_{z'})$. Then there exist $i', x, y \in \mathbb{N}$ with $i' > 0$ such that $c(i', x, y) \in Q'_{\kappa-1}(U_z)$ and $F_{i'}^{u_z}(x)$ asks the

query α. By the choice of κ, it holds $Q'_{\kappa-1}(U_{z'}) = Q'_{\kappa-1}(U_z)$ and thus $c(i', x, y) \in Q'_{\kappa-1}(U_{z'})$. Hence all queries of $F_{i'}^{u_{z'}}(x)$ are in $Q'_\kappa(U_{z'})$. However, $\alpha \notin Q'_\kappa(U_{z'})$ and therefore, α is not asked by $F_{i'}^{u_{z'}}(x)$. This shows that there is a word $\beta \in u_z \triangle u_{z'}$ asked by both $F_{i'}^{u_z}(x)$ and $F_{i'}^{u_{z'}}(x)$ (otherwise, the two computations would ask the same queries). But then $\beta \in Q'_\kappa(U_z) \cap Q'_\kappa(U_{z'})$, which implies that z and z' strongly conflict, a contradiction to the assumption that z and z' do not conflict. Hence $\kappa = 0$ and $U_z = Q'_0(U_z) \neq Q'_0(U_{z'}) = U_{z'}$.

Recall that U_z (resp., $U_{z'}$) is the set that consists of all oracle queries of $F_r^{u_z}(0^n)$ (resp., $F_r^{u_{z'}}(0^n)$) and all oracle queries of the least accepting path P (resp., P') of the computation $M_i^{u_z}(F_r^{u_z}(0^n))$ (resp., $M_i^{u_{z'}}(F_r^{u_{z'}}(0^n))$). As $u_z(q) = v'(q)$ for all $q \in Q(U_z) \supseteq U_z$ and $u_{z'}(q) = v'(q)$ for all $q \in Q(U_{z'}) \supseteq U_{z'}$, it holds that $F_r^{v'}(0^n) = F_r^{u_z}(0^n) = F_r^{u_{z'}}(0^n)$ and that the paths P and P' are accepting paths of the computation $M_i^{v'}(F_r^{v'}(0^n))$. Finally, P and P' are distinct paths, since U_z and $U_{z'}$ are distinct sets. This finishes the proof of (3.16) and thus also the proof of Claim 3.3.13. □

The remainder of the proof that the construction is possible is based on an idea by Hartmanis and Hemachandra [HH88]. Consider the set

$$\begin{aligned} E &= \{\{z, z'\} \mid z, z' \in \Sigma^n, z \text{ odd} \Leftrightarrow z' \text{ even}, (z \in Q(U_{z'}) \vee z' \in Q(U_z))\} \\ &\subseteq \bigcup_{z \in \Sigma^n} \{\{z, z'\} \mid z' \in \Sigma^n, z' \in Q(U_z)\}. \end{aligned} \tag{3.17}$$

Let $z, z' \in \Sigma^n$ such that (z odd $\Leftrightarrow$ z' even). Then by Claim 3.3.13 and Claim 3.3.10, z and z' conflict in a word of length n. As $u_z \triangle u_{z'} = \{z, z'\}$, this means that they conflict in z or z'. Hence $z \in Q(U_{z'})$ or $z' \in Q(U_z)$.

This shows $E = \{\{z, z'\} \mid z, z' \in \Sigma^n, z \text{ odd} \Leftrightarrow z' \text{ even}\}$ and thus $|E| = 2^{2n-2}$. By Claim 3.3.9, for each $z \in \Sigma^n$ it holds $|Q(U_z)| \leq \ell(Q(U_z)) \leq 2\gamma(n)$. Consequently,

$$|E| \overset{(3.17)}{\leq} \sum_{z \in \Sigma^n} |Q(U_z)| \leq 2^n \cdot 2\gamma(n) = 2^{n+1} \cdot \gamma(n) \overset{(3.13)}{<} 2^{2n-2} = |E|,$$

a contradiction. Hence the assumption made on page 60 —i.e., the assumption that for each odd $z \in \Sigma^n$ and each even $z' \in \Sigma^n$ the computations $M_i^{u_z}(F_r^{u_z}(0^n))$ and $M_j^{u_{z'}}(F_r^{u_{z'}}(0^n))$ definitely accept— leads to a contradiction. Thus the assumption that the construction fails in step s treating the task (i, j, r) is wrong. This shows that the construction described above is possible and O is well-defined. In order to finish the proof of Theorem 3.3.1, it remains to show that

- DisjNP^O does not contain a pair $\leq_{\mathrm{m}}^{\mathrm{p},O}$-hard for $\mathrm{UP}^O \cap \mathrm{coUP}^O$,
- each non-empty problem in NP^O has a P^O-optimal proof system, and
- UP^O does not contain a $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete problem.

Claim 3.3.14 DisjNP^O *does not contain a pair that is* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-hard for* $\mathrm{UP}^O \cap \mathrm{coUP}^O$.

Proof Assume the assertion is wrong, i.e., there exist distinct $i, j \in \mathbb{N}^+$ such that $(L(M_i^O), L(M_j^O)) \in \mathrm{DisjNP}^O$ and for every $A \in \mathrm{UP}^O \cap \mathrm{coUP}^O$ it holds $A \leq_{\mathrm{m}}^{\mathrm{p},O} (L(M_i^O), L(M_j^O))$. From $L(M_i^O) \cap L(M_j^O) = \emptyset$ if follows that for all s there does not exist z such that both $M_i^{w_s}(z)$ and $M_j^{w_s}(z)$ definitely accept. Hence by V2, there does not exist s with $t_s(i, j) = 0$ and thus by construction, $t_s(i, j) = -p$ for some $p \in \mathbb{P}_{\equiv 3}$ and all sufficiently large s. The latter implies $|O \cap \Sigma^{p^k}| = 1$ for all $k > 0$ (cf. V3), which yields $A_p^O = \overline{B_p^O}$ and $A_p^O \in \mathrm{UP}^O \cap \mathrm{coUP}^O$. Thus there exists r such that $A_p^O \leq_{\mathrm{m}}^{\mathrm{p},O} (L(M_i^O), L(M_j^O))$ via F_r^O. Let s be the step that treats task (i, j, r). This step makes sure that there exists $n \in \mathbb{N}^+$ such that at least one of the following properties holds:

- $\forall_{v \sqsupseteq w_s} 0^n \in A_p^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
- $\forall_{v \sqsupseteq w_s} 0^n \in B_p^v$, $F_r^{w_s}(0^n)$ is definite, and $M_j^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

As $O(q) = w_s(q)$ for all q that w_s is defined for, one of the following two statements holds.

- $0^n \in A_p^O$ and $F_r^O(0^n)$ is rejected by M_i^O.
- $0^n \in B_p^O = \overline{A_p^O}$ and $F_r^O(0^n)$ is rejected by M_j^O.

This is a contradiction to $A_p^O \leq_{\mathrm{m}}^{\mathrm{p},O} (L(M_i^O), L(M_j^O))$ via F_r^O, which completes the proof of Claim 3.3.14. □

Claim 3.3.15 *Each non-empty problem in* NP^O *has a* P^O*-optimal proof system.*

Proof By Corollary 2.3.6, it suffices to prove that $\mathcal{CAN}^O$ has a P^O-optimal proof system.

Let $g \in \mathrm{FP}^O$ be an arbitrary proof system for $\mathcal{CAN}^O$ and a be an arbitrary element of $\mathcal{CAN}^O$. Define f to be the following function $\Sigma^* \to \Sigma^*$:

$$f(z) = \begin{cases} g(z') & \text{if } z = 1z' \\ y & \text{if } z = 0c(i,x,y) \text{ for } i \in \mathbb{N}^+,\ x, y \in \mathbb{N}, \text{ and } c(i,x,y) \in O \\ a & \text{otherwise} \end{cases}$$

By definition, $f \in \mathrm{FP}^O$ and as g is a proof system for $\mathcal{CAN}^O$ it holds $f(\Sigma^*) \supseteq \mathcal{CAN}^O$. We show $f(\Sigma^*) \subseteq \mathcal{CAN}^O$. Let $z \in \Sigma^*$. Assume $z = 0c(i,x,y)$ for $i \in \mathbb{N}^+$, $x, y \in \mathbb{N}$, and $c(i,x,y) \in O$ (otherwise, clearly $f(z) \in \mathcal{CAN}^O$). Let s be large enough such that w_s is defined for $c(i,x,y)$, i.e., $w_s(c(i,x,y)) = 1$. As w_s is t_s-valid, we obtain by V1 that $y \in \mathcal{CAN}^{w_s}$ and by Claim 3.3.4 that $y \in \mathcal{CAN}^v$ for all $v \sqsupseteq w_s$. Thus $y \in \mathcal{CAN}^O$, which shows that f is a proof system for $\mathcal{CAN}^O$.

It remains to show that each proof system for $\mathcal{CAN}^O$ is P^O-simulated by f. Let h be an arbitrary proof system for $\mathcal{CAN}^O$. Then there exists $i > 0$ such that F_i^O computes h. By construction, $t_s(i) > 0$ for s being the number of the step that treats the task i. Consider the following function $\pi\colon \Sigma^* \to \Sigma^*$:

$$\pi(x) = \begin{cases} 0c(i,x,F_i^O(x)) & \text{if } c(i,x,F_i^O(x)) \geq t_s(i) \\ z & \text{if } c(i,x,F_i^O(x)) < t_s(i) \text{ and } z \text{ is minimal with } f(z) = F_i^O(x) \end{cases}$$

As f and F_i^O are proof systems for $\mathcal{CAN}^O$, for every x there exists z with $f(z) = F_i^O(x)$. Hence π is total. Since $t_s(i)$ is a constant, $\pi \in \mathrm{FP}^O$. It remains to show that $f(\pi(x)) = F_i^O(x)$ for all $x \in \Sigma^*$. If $c(i,x,F_i^O(x)) < t_s(i)$, it holds $f(\pi(x)) = F_i^O(x)$. Otherwise, choose s' large enough such that (i) $t_{s'}(i)$ is defined (i.e., $t_{s'}(i) = t_s(i) > 0$) and (ii) $w_{s'}$ is defined for $c(i,x,F_i^{w_{s'}}(x))$. Then, as $w_{s'}$ is $t_{s'}$-valid, V5 yields that $c(i,x,F_i^{w_{s'}}(x)) \in w_{s'}$. By Claim 3.3.4, $F_i^{w_{s'}}(x)$ is definite and hence $F_i^O(x) = F_i^{w_{s'}}(x)$ as well as $c(i,x,F_i^O(x)) \in w_{s'} \subseteq O$. Thus $f(\pi(x)) = F_i^O(x)$. □

Claim 3.3.16 UP^O *does not contain a* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problem.*

Proof Assume there exists an UP^O-complete problem. Then there exists $i > 0$ such that $L(M_i^O)$ is $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete for UP^O. According to Remark 3.3.3 we may assume without loss of generality that on every input, M_i^O has at most one accepting path. Thus there does not exist

$s > 0$ with $t_s(i,i) = 0$ (cf. V6). Hence by construction, $t_s(i,i) = -q$ for some $q \in \mathbb{P}_{\equiv 1}$ and all sufficiently large s. Then $|O \cap \Sigma^{q^k}| \leq 1$ for all $k > 0$ (cf. V7) and consequently, $C_q^O \in \mathrm{UP}^O$. As $L(M_i^O)$ is complete for UP^O, there exists $r > 0$ such that $C_q^O \leq_{\mathrm{m}}^{\mathrm{p},O} L(M_i^O)$ via F_r^O. Let $s > 0$ be the step that treats the task (i,i,r). By construction, there exists $n \in \mathbb{N}^+$ such that one of the following two statements holds:

- $\forall_{v \sqsupseteq w_s}\, 0^n \in C_q^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
- $\forall_{v \sqsupseteq w_s}\, 0^n \notin C_q^v$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely accepts.

As O and w_s agree on all words that w_s is defined for, one of the following two conditions holds:

- $0^n \in C_q^O$ and $M_i^O(F_r^O(0^n))$ rejects.
- $0^n \notin C_q^O$ and $M_i^O(F_r^O(0^n))$ accepts.

This is a contradiction to $C_q^O \leq_{\mathrm{m}}^{\mathrm{p},O} L(M_i^O)$ via F_r^O, which shows that UP^O does not have $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete problems. This completes the proof of Claim 3.3.16. □

Now the proof of Theorem 3.3.1 is complete. □

3.4 NP ∩ coNP and ¬CON Relative to an Oracle

In this section we construct another oracle which Pudlák [Pud17] asks for. It proves that the relativized conjectures CON and NP ∩ coNP are different. Putting it more precisely, it holds that NP ∩ coNP does not contain problems that are hard for UP ∩ coUP and that all non-empty sets in coNP have P-optimal proof systems relative to the oracle. Hence in particular, it holds NP ∩ coNP ∧ ¬CON relative to the oracle and this implies that there does not exist a relativizable proof for the implication NP ∩ coNP ⇒ CON.

As Figure 1.2 illustrates, this implies that also the following implications do not admit relativizable proofs: NP ∩ coNP ⇒ $\mathsf{CON}^{\mathsf{N}}$, NP ∩ coNP ⇒ DisjNP, and NP ∩ coNP ⇒ UP, where an oracle relative to which the last implication does not holds was already known by Corollary 3.2.4.

For all of the four implications mentioned in the previous paragraphs there recently have been constructed oracles by Fabian Egidy, Anton Ehrmanntraut, and Christian Glaßer [unpublished, private communication] that show that also the converse implications cannot be proven using exclusively relativizable proof techniques.

Finally, as a corollary we obtain that additionally there do not exist NP-complete problems having P-optimal proof systems relative to the oracle, i.e., we even have a relativized world in which NP ∩ coNP ∧ SAT ∧ ¬CON holds. However, this does not yield a further new separation in Pudlák's research program, since an oracle relative to which SAT ∧ ¬CON was already known before [Kha19].

Theorem 3.4.1 *There exists an oracle O such that the following statements hold:*

- $\mathrm{NP}^O \cap \mathrm{coNP}^O$ *does not contain problems that are* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-hard for* $\mathrm{UP}^O \cap \mathrm{coUP}^O$.
- $\overline{\mathcal{CAN}}^O$ *has* P^O*-optimal proof systems.*

Proof Let D be a (possibly partial) oracle and $p \in \mathbb{P}_{\geq 3}$. We define

$$\begin{aligned} A_p^D &:= \{0^{p^k} \mid k \in \mathbb{N}^+, \exists_{x \in \Sigma^{p^k}} x \in D \text{ and } x \text{ odd}\} \cup \overline{\{0^{p^k} \mid k \in \mathbb{N}^+\}} \\ B_p^D &:= \{0^{p^k} \mid k \in \mathbb{N}^+, \exists_{x \in \Sigma^{p^k}} x \in D \text{ and } x \text{ even}\} \end{aligned}$$

Note that if $|\Sigma^{p^k} \cap D| = 1$ for each $k \in \mathbb{N}^+$, then $A_p^D, B_p^D \in \mathrm{UP}^D$ and $A_p^D = \overline{B_p^D}$, which implies $A_p^D \in \mathrm{UP}^D \cap \mathrm{coUP}^D$. For all relevant $p \in \mathbb{P}_{\geq 3}$ our construction will ensure that $|\Sigma^{p^k} \cap O| = 1$ for each $k \in \mathbb{N}^+$ relative to the final oracle O.

For the sake of simplicity, let us call a pair (M_i, M_j) an $\mathrm{NP}^D \cap \mathrm{coNP}^D$-machine if $L(M_i^D) = \overline{L(M_j^D)}$. Note that throughout this proof we sometimes omit the oracles in the superscript, e.g., we write NP or A_p instead of NP^D or A_p^D. However, we do not do that in the "actual" proof but only when explaining ideas in a loose way in order to convey the intuition behind the occasionally technical arguments.

Preview of the Construction We sketch the basic ideas of our construction.

1. For all $i > 0$ we try to ensure that F_i is not a proof system for $\overline{\mathcal{CAN}}$ relative to the final oracle. If this is possible, we do not have to consider F_i anymore. If it is not possible, then F_i inherently is a proof system for $\overline{\mathcal{CAN}}$. In that case we start to encode the values of F_i into the oracle. This way we easily obtain a P-optimal proof system for $\overline{\mathcal{CAN}}$ in the end. Note that it is crucial that we also allow to encode values of functions F_j into the oracle before we try —as described above— to make sure that these functions are not proof systems for $\overline{\mathcal{CAN}}$. Hence the final oracle may also contain encodings of values of functions that are not proof systems for $\overline{\mathcal{CAN}}$.

2. Similarly, for each pair (i, j) with $i \neq j$ we first try to make sure that (M_i, M_j) is not an $\mathrm{NP} \cap \mathrm{coNP}$-machine. If this is possible, then we need not consider the pair (M_i, M_j) anymore. If it is not possible, then (M_i, M_j) inherently is an $\mathrm{NP} \cap \mathrm{coNP}$-machine. In this case we choose a prime p and ensure in the further construction that $A_p = \overline{B_p}$ and $A_p \in \mathrm{UP} \cap \mathrm{coUP}$. Moreover, we diagonalize against all FP-functions F_r in order to make sure that F_r does not reduce A_p to $L(M_i)$.

For $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ we write $c(i, x, y) := \langle 0^i, 0^{|x|^i + i}, 0^{|x|^i + i}, x, y, y \rangle$. Note that by the properties of the pairing function $\langle \cdot \rangle$, $|c(i, x, y)|$ is even and

$$\forall_{i \in \mathbb{N}^+, x, y \in \mathbb{N}} \; |c(i, x, y)| > 4 \cdot \max(|x|^i + i, |y|). \tag{3.18}$$

The following claim can be proven in the same way as Claim 3.3.4. The only difference is that we use a slightly different coding function c here.

Claim 3.4.2 ([DG19]) *Let $w \in \Sigma^*$, $i \in \mathbb{N}^+$, and $x, y \in \mathbb{N}$ such that $c(i, x, y) \leq |w|$. Then the following holds.*

1. *$F_i^w(x)$ is definite and $F_i^w(x) < |w|$.*

2. *For all $v \sqsupseteq w$, it holds $(F_i^w(x) \in \mathcal{CAN}^w \Leftrightarrow F_i^w(x) \in \mathcal{CAN}^v)$.*

During the construction we maintain a growing collection of requirements that is represented by a partial function belonging to the set

$$\mathcal{T} = \Big\{ t\colon (\mathbb{N}^+)^2 \to \mathbb{Z} \mid \operatorname{dom}(t) \text{ is finite, } t \text{ is injective on its support, and}$$
$$\bullet\ t(\{(i,i) \mid i \in \mathbb{N}^+\}) \subseteq \{0\} \cup \mathbb{N}^+$$
$$\bullet\ t(\{(i,j) \mid i,j \in \mathbb{N}^+, i \neq j\}) \subseteq \{0\} \cup \{-p \mid p \in \mathbb{P}_{\geq 3}\} \Big\}.$$

A partial oracle $w \in \Sigma^*$ is called t-valid for $t \in \mathcal{T}$ if it satisfies the following properties.

V1 For all $i \in \mathbb{N}^+$ and all $x, y \in \mathbb{N}$, if $c(i,x,y) \in w$, then $F_i^w(x) = y$ and $y \in \overline{\mathcal{CAN}^w}$.
(meaning: if the oracle contains the codeword $c(i,x,y)$, then $F_i^w(x)$ outputs y and $y \in \overline{\mathcal{CAN}^w}$; hence $c(i,x,y) \in w$ is a proof for $y \in \overline{\mathcal{CAN}^w}$)

V2 For all distinct $i, j \in \mathbb{N}^+$, if $t(i,j) = 0$, then there exists x such that (i) $M_i^w(x)$ and $M_j^w(x)$ definitely accept or (ii) $M_i^w(x)$ and $M_j^w(x)$ definitely reject.
(meaning: for every extension of the oracle, (M_i, M_j) is not an NP ∩ coNP-machine.)

V3 For all distinct $i, j \in \mathbb{N}^+$ with $t(i,j) = -p$ for some $p \in \mathbb{P}_{\geq 3}$ and each $k \in \mathbb{N}^+$, it holds

(i) $|\Sigma^{p^k} \cap w| \leq 1$ and

(ii) if w is defined for all words of length p^k, then $|\Sigma^{p^k} \cap w| \geq 1$.

(meaning: if $t(i,j) = -p$, then ensure that $A_p = \overline{B_p}$ and $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ relative to the final oracle.)

V4 For all $i \in \mathbb{N}^+$ with $t(i,i) = 0$, there exists x such that $F_i^w(x)$ is definite and $F_i^w(x) \in \mathcal{CAN}^v$ for all $v \sqsupseteq w$.
(meaning: for every extension of the oracle, F_i is not a proof system for $\overline{\mathcal{CAN}}$)

V5 For all $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t(i,i) \leq c(i,x,F_i^w(x)) < |w|$, it holds $c(i,x,F_i^w(x)) \in w$.
(meaning: if $t(i) > 0$, then from $t(i)$ on, we encode F_i into the oracle.
Note that V5 is not in contradiction with V3 as $|c(\cdot,\cdot,\cdot)|$ is even.)

The subsequent claim follows directly from the definition of t-valid.

Claim 3.4.3 *Let $t, t' \in \mathcal{T}$ such that t' is an extension of t. For all oracles $w \in \Sigma^*$, if w is t'-valid, then w is t-valid.*

Claim 3.4.4 *Let $t \in \mathcal{T}$ and $u, v, w \in \Sigma^*$ such that $u \sqsubseteq v \sqsubseteq w$ and both u and w are t-valid. Then v is t-valid.*

Proof The oracle v satisfies V2 and V4, since u satisfies these conditions. Moreover, v satisfies V3 as w satisfies these conditions.

Let $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ such that $c(i,x,y) \in v$. Then $c(i,x,y) \in w$ and as w is t-valid, we obtain by V1 that $F_i^w(x) = y \in \overline{\mathcal{CAN}^w}$. Claim 3.4.2 yields that $F_i^v(x)$ is definite and $F_i^v(x) \in \mathcal{CAN}^v \Leftrightarrow F_i^v(x) \in \mathcal{CAN}^w$. This yields $F_i^v(x) = F_i^w(x) = y \in \overline{\mathcal{CAN}^v}$. Thus v satisfies V1.

Now let $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ such that $0 < t(i,i) \leq c(i,x,F_i^v(x)) < |v|$. Again, by Claim 3.4.2, $F_i^v(x)$ is definite and thus $F_i^v(x) = F_i^w(x)$. As $|v| \leq |w|$ and w is t-valid, we obtain by V5

that $c(i,x,F_i^v(x)) = c(i,x,F_i^w(x)) \in w$. Since $v \sqsubseteq w$ and $|v| > c(i,x,F_i^v(x))$, we obtain $c(i,x,F_i^v(x)) \in v$, which shows that v satisfies V5. □

Oracle construction. Let T be an injective enumeration of $(\mathbb{N}^+)^2 \cup \{(i,j,r) \in (\mathbb{N}^+)^3 \mid i \neq j\}$ having the property that for all $i,j,r \in \mathbb{N}^+$ with $i \neq j$ the pair (i,j) appears earlier than the triple (i,j,r). Each element of T stands for a task. We treat the tasks in the order specified by T and after treating a task we remove it and possibly other tasks from T.

We begin with the unique nowhere defined function $t_0 \in \mathcal{T}$ and the unique $w_0 \in \Sigma^*$ with $|w_0| = 1$ and —when considering w_0 as a set— with $w_0 = \emptyset$ (i.e., $w_0 = 0$ when we consider w_0 as a word in Σ^* and $w_0 = 1$ when we consider w_0 as a natural number; cf. the paragraph "Identification of Σ^* and $\mathbb{N}$" on page 31). Observe that w_0 is t_0-valid. Then we start treating the tasks in T. Each task will be treated by (strictly) extending the corresponding oracle w_i to w_{i+1} and adding further requirements to the respective "valid function" t_i in order to obtain an extension t_{i+1} of t_i. Thus we define partial functions $t_1, t_2, \ldots$ in $\mathcal{T}$ and partial oracles $w_0 \sqsubsetneq w_1 \sqsubsetneq w_2 \sqsubsetneq \ldots$ such that each t_{i+1} is an extension of t_i and each w_i is t_i-valid.

Finally, we choose $O = \bigcup_{i=0}^{\infty} w_i$. Thus O is totally defined, since in each step we strictly extend the oracle.

Let us describe step $s > 0$, which starts with some $t_{s-1} \in \mathcal{T}$ and a t_{s-1}-valid oracle w_{s-1} and chooses an extension $t_s \in \mathcal{T}$ of t_{s-1} and a t_s-valid $w_s \supsetneq w_{s-1}$ (it will be argued later that the construction described below is indeed possible). Let us recall that each task is immediately deleted from T after it is treated.

- task (i,i): Let $t' = t_{s-1} \cup \{(i,i) \mapsto 0\}$. If there exists a t'-valid $v \supsetneq w_{s-1}$, then let $t_s = t'$ and w_s be the least t'-valid, partial oracle $\supsetneq w_{s-1}$. Otherwise, let $t_s = t_{s-1} \cup \{(i,i) \mapsto |w_{s-1}|\}$(note that here $|w_{s-1}| > 0$ since $|w_0| = 1$[5]) and choose $b \in \{0,1\}$ and $w_s = w_{s-1}b$ such that w_s is t_s-valid.
  (meaning: try to ensure that F_i is not a proof system for $\overline{\mathcal{CAN}}$. If this is impossible, require that from now on the values of F_i are encoded into the oracle.)

- task (i,j) with $i \neq j$: Let $t' = t_{s-1} \cup \{(i,j) \mapsto 0\}$. If there exists a t'-valid $v \supsetneq w_{s-1}$, then let $t_s = t'$, define w_s to be the least t'-valid, partial oracle $\supsetneq w_{s-1}$, and delete all tasks $(i,j,\cdot)$ from T. Otherwise, let $z = |w_{s-1}|$, choose some $p \in \mathbb{P}_{\geq 3}$ greater than $|z|$ with $-p \notin \operatorname{ran}(t_{s-1})$, let $t_s = t_{s-1} \cup \{(i,j) \mapsto -p\}$, and choose $b \in \{0,1\}$ and $w_s = w_{s-1}b$ such that w_s is t_s-valid.
  (meaning: try to ensure that (M_i, M_j) is not an NP ∩ coNP-machine. If this is impossible, then $L(M_i)$ inherently is in NP∩coNP, we choose a sufficiently large prime p, and ensure by the choice $t_s(i,j) = -p$ that relative to the final oracle it holds $A_p = \overline{B_p}$ and $A_p, B_p \in \mathrm{UP}$, which implies $A_p \in \mathrm{UP} \cap \mathrm{coUP}$.)

- task (i,j,r) with $i \neq j$: It holds $t_{s-1}(i,j) = -p$ for a prime $p \in \mathbb{P}_{\geq 3}$, since otherwise, this task would have been deleted in the treatment of task (i,j). Define $t_s = t_{s-1}$ and choose a t_s-valid $w_s \supsetneq w_{s-1}$ such that there is some $n \in \mathbb{N}^+$ such that one of the following two statements holds:

  – $0^n \in A_p^{w_s}$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
  – $0^n \in B_p^{w_s}$, $F_r^{w_s}(0^n)$ is definite, and $M_j^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

  Note: As w_s is defined for all words of length n, the statement $0^n \in A_p^{w_s}$ (resp., $0^n \in B_p^{w_s}$) is equivalent with the statement $\forall_{v \sqsupseteq w_s}\, 0^n \in A_p^v$ (resp., $\forall_{v \sqsupseteq w_s}\, 0^n \in B_p^v$).

[5] Indeed, this is the reason why we do not start with $w_0 = \varepsilon$.

(meaning: due to V3 it will hold $A_p \in \mathrm{UP} \cap \mathrm{coUP}$ relative to the final oracle. As $t_{s-1}(i,j) < 0$, it will hold $L(M_i^O) = \overline{L(M_j^O)}$ and hence $L(M_i^O) \in \mathrm{NP}^O$. Thus the treatment of the task (i,j,r) makes sure that F_r does not reduce A_p to $L(M_i)$ relative to the final oracle.)

Observe inductively that each t_s is in $\mathcal{T}$. We now show that the construction is possible. For that purpose, we first describe how a valid oracle can be extended by one bit such that it remains valid.

Claim 3.4.5 *Let $s \in \mathbb{N}$ and $w \in \Sigma^*$ be a t_s-valid oracle with $w \sqsupseteq w_s$. Moreover, let $z = |w|$. Then the following statements hold.*

1. *If $|z|$ is odd and for all $p \in \mathbb{P}_{\geq 3}$ and $k \in \mathbb{N}^+$ with $-p \in \mathrm{ran}(t_s)$ it holds $|z| \neq p^k$, then $w0$ and $w1$ are t_s-valid.*
2. *If there exist $p \in \mathbb{P}_{\geq 3}$ and $k \in \mathbb{N}^+$ with $-p \in \mathrm{ran}(t_s)$ such that $|z| = p^k$, $z \neq 1^{p^k}$, and $w \cap \Sigma^{p^k} = \emptyset$, then $w0$ and $w1$ are t_s-valid.*
3. *If there exist $p \in \mathbb{P}_{\geq 3}$ and $k \in \mathbb{N}^+$ with $-p \in \mathrm{ran}(t_s)$ such that $z = 1^{p^k}$ and $w \cap \Sigma^{p^k} = \emptyset$, then $w1$ is t_s-valid.*
4. *If $z = c(i, x, F_i^w(x))$ for $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ and $0 < t_s(i,i) \leq z$, then $w1$ is t_s-valid.*
5. *If $z = c(i, x, F_i^w(x))$ for $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$, at least one of the three conditions (i) $t_s(i,i)$ undefined, (ii) $t_s(i,i) = 0$, and (iii) $t_s(i,i) > z$ holds, and $F_i^w(x) \in \overline{\mathcal{CAN}^w}$, then $w0$ and $w1$ are t_s-valid.*
6. *In all other cases (i.e., none of the assumptions in 1–5 holds) $w0$ is t_s-valid.*

Proof First note that V2 and V4 are not affected by extending the oracle. So we only need to consider V1, V3, and V5 in the following.

Let us show the following assertions.

$w0$ satisfies V1. (3.19)

If (i) $z = c(i, x, F_i^w(x))$ for $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $F_i^w(x) \in \overline{\mathcal{CAN}^w}$ or (ii) z has odd length, then $w1$ satisfies V1. (3.20)

$w0$ satisfies V5 unless there exist $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ such that (i) $z = c(i,x,y)$, (ii) $0 < t_s(i,i) \leq z$, and (iii) $F_i^w(x) = y$. (3.21)

$w1$ satisfies V5. (3.22)

(3.19) and (3.20): Let $i' \in \mathbb{N}^+$ and $x', y' \in \mathbb{N}$ such that $c(i', x', y') \in w$. Then, as w is t_s-valid, by V1, $F_{i'}^w(x') = y' \in \overline{\mathcal{CAN}^w}$ and by Claim 3.4.2, $F_{i'}^w(x')$ is definite and $y' \in \overline{\mathcal{CAN}^v}$ for all $v \sqsupseteq w$. Hence in particular, $F_{i'}^{wb}(x') = y' \in \overline{\mathcal{CAN}^{wb}}$ for all $b \in \{0,1\}$. This shows (3.19).

For the proof of (3.20) it remains to consider z. In case (ii) $w1$ satisfies V1 as $|z|$ is odd and $c(\cdot,\cdot,\cdot)$ has even length. Consider case (i), i.e., $z = c(i, x, F_i^w(x))$ for $i \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $F_i^w(x) \in \overline{\mathcal{CAN}^w}$. Then by Claim 3.4.2, $F_i^{w1}(x) = F_i^w(x) \in \overline{\mathcal{CAN}^{w1}}$, which shows that $w1$ satisfies V1 and thus finishes the proof of (3.20).

(3.21) and (3.22): Let $i' \in \mathbb{N}^+$ and $x' \in \mathbb{N}$ such that $0 < t_s(i',i') \leq c(i', x', F_{i'}^w(x')) < |w|$. Then by Claim 3.4.2, $F_{i'}^w(x')$ is definite and thus $F_{i'}^{wb}(x') = F_{i'}^w(x')$ for all $b \in \{0,1\}$. As w is t_s-valid, we have $c(i', x', F_{i'}^w(x')) \in w$ by V5 and hence $c(i', x', F_{i'}^{wb}(x')) \in w \subseteq wb$ for all $b \in \{0,1\}$. This shows (3.22).

In order to finish the proof of (3.21), it remains to consider z. Assume $z = c(i, x, y)$ for some $i, x, y \in \mathbb{N}$ with $i > 0$ and $0 < t_s(i, i) \leq z$ (otherwise, $w0$ satisfies V5). If (iii) is wrong, then $F_i^w(x) \neq y$. By Claim 3.4.2, this computation is defined and hence $F_i^{w0}(x) \neq y$, which proves that $w0$ satisfies V5. This shows (3.21).

We now prove the statements 1–6.

1. Observe that $w0$ and $w1$ satisfy V3. Moreover, by (3.19) and (3.21), the oracle $w0$ satisfies V1 and V5 (recall that the length of each $c(\cdot, \cdot, \cdot)$ is even). By (3.20) and (3.22), the oracle $w1$ satisfies V1 and V5.

2. By (3.19), (3.20), (3.21), and (3.22), the oracles $w0$ and $w1$ satisfy V1 and V5. As $z \neq 1^{p^k}$ and w satisfies V3, the oracle $w0$ satisfies V3. As $w \cap \Sigma^{p^k} = \emptyset$, the oracle $w1$ satisfies V3.

3. By (3.20) and (3.22), the oracle $w1$ satisfies V1 and V5. As $w \cap \Sigma^{p^k} = \emptyset$, the oracle $w1$ satisfies V3.

4. As $|z|$ is even, $w1$ satisfies V3. By (3.22), $w1$ satisfies V5. It remains to argue that $w1$ satisfies V1. In order to apply (3.20), which will yield that $w1$ satisfies V1, it is sufficient to prove $y := F_i^w(x) \in \overline{\mathcal{CAN}^w}$. For a contradiction assume $y \in \mathcal{CAN}^w$. Let $s' > 0$ be the step that treats the task (i, i). Note $s' \leq s$ since $t_s(i, i)$ is defined. By Claim 3.4.3, w is $t_{s'-1}$-valid. As by Claim 3.4.2 the computation $F_i^w(x)$ is definite and $y \in \mathcal{CAN}^v$ for all $v \sqsupseteq w$, the oracle w is even t-valid for $t = t_{s'-1} \cup \{(i, i) \mapsto 0\}$. But then the construction would have chosen $t_{s'} = t$, in contradiction to $t_s(i, i) > 0$.

5. As $|z|$ is even, $w0$ and $w1$ satisfy V3. By (3.19), (3.20), and (3.22), $w0$ satisfies V1 and $w1$ satisfies both V1 and V5. Moreover, (3.21) can be applied, since each of the conditions (i)–(iii) of statement 5 implies that condition (ii) of (3.21) does not hold. Thus $w0$ satisfies V5.

6. By (3.19), $w0$ satisfies V1. If $w0$ does not satisfy V3, then there exist $p \in \mathbb{P}_{\geq 3}$ with $-p \in \mathrm{ran}(t_s)$ and $k > 0$ such that $w \cap \Sigma^{p^k} = \emptyset$ and $z = 1^{p^k}$, but this case is covered by statement 3 of the current claim. If $w0$ does not satisfy V5, then by (3.21), there exist $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ such that (i) $z = c(i, x, y)$, (ii) $0 < t_s(i, i) \leq z$, and (iii) $F_i^w(x) = y$. This case, however, is covered by statement 4 of the current claim.

This finishes the proof of Claim 3.4.5. □

In order to show that the above construction is possible, assume that it is not possible and let $s > 0$ be the least number for which it fails.

If step s treats a task $(i, j) \in (\mathbb{N}^+)^2$, then $t_{s-1}(i, j)$ is not defined, since the value of the "valid function" at the point (i, j) is defined in the unique treatment of the task (i, j). Hence $t' = t_{s-1} \cup \{(i, j) \mapsto 0\}$ is well-defined. If t_s is chosen to be t', then the construction clearly is possible. Otherwise, due to the sufficiently large choice of $t_s(i, j)$, the t_{s-1}-valid oracle w_{s-1} is even t_s-valid and Claim 3.4.5 ensures that there exists a t_s-valid $w_{s-1}b$ for some $b \in \{0, 1\}$. Hence the construction does not fail in step s, a contradiction.

For the remainder of the proof that the construction above is possible we assume that step s treats a task $(i, j, r) \in (\mathbb{N}^+)^3$ with $i \neq j$.

Then $t_s = t_{s-1}$ and $t_s(i, j) = -p$ for some $p \in \mathbb{P}_{\geq 3}$ (recall that otherwise, the task (i, j, r) would have been deleted in the step treating the task (i, j)). Let γ be the polynomial defined by $x \mapsto (x^r + r)^{i+j} + i + j$ and choose $k \in \mathbb{N}^+$ such that for $n = p^k$ it holds

$$2^{n-1} > 2 \cdot \gamma(n) \tag{3.23}$$

and w_{s-1} is undefined for all words of length $\geq n$. Note that by the choice of γ, for all oracles D, all oracle queries asked by the computations $F_r^D(0^n)$, $M_i^D(F_r^D(0^n))$, and $M_j^D(F_r^D(0^n))$ are of length $\leq \gamma(n)$.

We define $u \sqsupseteq w_{s-1}$ to be the minimal t_s-valid oracle that is defined for all words of length $< n$. Such an oracle exists by Claim 3.4.5.

Moreover, for $z \in \Sigma^n$, let $u_z \sqsupsetneq u$ be the minimal t_s-valid oracle with $u_z \cap \Sigma^n = \{z\}$ that is defined for all words of length $\leq \gamma(n)$. Such an oracle exists by Claim 3.4.5: first, starting from u we extend the current oracle bitwise such that (i) it remains t_s-valid, (ii) it is defined for precisely the words of length $\leq n$, and (iii) its intersection with Σ^n equals $\{z\}$. This is possible by 2, 3, and 6 of Claim 3.4.5. Then by Claim 3.4.5, the current oracle can be extended bitwise without losing its t_s-validity until it is defined for all words of length $\leq \gamma(n)$.

We define a further oracle v that will be crucial in the following. Let $s' > 0$ be the step that treats the task (i, j). As $t_s(i, j)$ is defined, it holds $s' \leq s$. By Claim 3.4.3, the oracle u is $t_{s'-1}$-valid. In order to define v, we need the following two properties (3.24) and (3.25). But first we define $\widetilde{u}$ to be the minimal partial oracle $\sqsupseteq u$ that is defined for all words of length $\leq n$, i.e., $\widetilde{u} = u$ when considering the oracles as sets. By Claim 3.4.5.1 and as $t_{s'-1}$ is not defined for the pair (i, j) and thus $-p \notin \operatorname{ran}(t_{s'-1})$, we obtain that $\widetilde{u}$ is $t_{s'-1}$-valid. Now let $w \sqsupseteq \widetilde{u}$ be a $t_{s'-1}$-valid oracle. We say that w satisfies property (3.24) if

$$\text{for all } i', x \in \mathbb{N} \text{ with } i' > 0,\ t_s(i', i') > 0, \text{ and } |\widetilde{u}| \leq c(i', x, F_{i'}^w(x)) < |w|, \text{ if } F_{i'}^w(x) \in \overline{\mathcal{CAN}^w}, \text{ then } c(i', x, F_{i'}^w(x)) \in w. \tag{3.24}$$

Note that by construction, for all $i' > 0$, if $t_s(i', i') > 0$, then $t_s(i', i') \leq |\widetilde{u}|$. Moreover, w satisfies property (3.25) if

$$\begin{aligned} &\text{for all } p' \in \mathbb{P}_{>3} \text{ with } -p' \in \operatorname{ran}(t_s), \text{ it holds} \\ &\quad \text{(i)}\ \ w \cap \Sigma^{p'^\kappa} \subseteq \{1^{p'^\kappa}\} \text{ for all } \kappa > 0 \text{ with } n < p'^\kappa \text{ and} \\ &\quad \text{(ii)}\ \ w \cap \Sigma^{p'^\kappa} = \{1^{p'^\kappa}\} \text{ for all } \kappa > 0 \text{ with } n < p'^\kappa \text{ and } 1^{p'^\kappa} < |w|. \end{aligned} \tag{3.25}$$

So roughly speaking, a $t_{s'-1}$-valid oracle $w \sqsupseteq \widetilde{u}$ satisfies properties (3.24) and (3.25) if it contains possibly all encodings of length $> n$ that we would demand a t_s-valid oracle of the same length to have.

Now we define $v \sqsupsetneq \widetilde{u}$ to be the minimal $t_{s'-1}$-valid oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies properties (3.24) and (3.25). Let us argue that such an oracle exists. Clearly $\widetilde{u}$ satisfies properties (3.24) and (3.25). Hence the second statement of the following claim shows that v is well-defined.

Claim 3.4.6 *1. For all $t_{s'-1}$-valid oracles w and w' with $\widetilde{u} \sqsubseteq w \sqsubseteq w'$, if w satisfies property (3.24) and w' does not satisfy property (3.24), then there exists $\alpha \in [|w|, |w'|)$ such that (i) $\alpha = c(i', x, F_{i'}^{w'}(x))$ for $i', x \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$, (ii) $F_{i'}^{w'}(x) \in \overline{\mathcal{CAN}^{w'}}$, and (iii) $\alpha \notin w'$.*

2. For each $t_{s'-1}$-valid oracle $w \sqsupseteq \widetilde{u}$ that satisfies properties (3.24) and (3.25) there exists $b \in \{0, 1\}$ such that wb is $t_{s'-1}$-valid and satisfies properties (3.24) and (3.25).

3. Let $w \sqsupseteq \widetilde{u}$ be $t_{s'-1}$-valid. If w satisfies properties (3.24) and (3.25), then each w' with $\widetilde{u} \sqsubseteq w' \sqsubseteq w$ satisfies properties (3.24) and (3.25) as well.

Proof 1. Since w' does not satisfy property (3.24), there exists $\alpha = c(i', x, F_{i'}^{w'}(x)) \in [|\widetilde{u}|, |w'|)$ for some $i', x \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$ such that $F_{i'}^{w'}(x) \in \overline{\mathcal{CAN}^{w'}}$ and $\alpha \notin w'$. For a contradiction, we assume $\alpha < |w|$. Then Claim 3.4.2 yields $F_{i'}^{w}(x) = F_{i'}^{w'}(x) \in \overline{\mathcal{CAN}^{w}}$. It

follows from $\alpha < |w|$, $w \sqsubseteq w'$, and $\alpha \notin w'$ that $\alpha \notin w$, which contradicts the assumption that w satisfies property (3.24). Hence $\alpha \geq |w|$, which proves statement 1.

2. We study several cases depending on $\alpha = |w|$ (i.e., α is the least word that w is not defined for).

- If α is of the form $c(i', x, F_{i'}^{w}(x))$ for $i', x \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$ such that $F_{i'}^{w}(x) \in \overline{\mathcal{CAN}^w}$, then we choose $b = 1$. The statements 4 and 5 of Claim 3.4.5 state that the oracle wb is $t_{s'-1}$-valid (recall that by construction $t_s(i', i') \leq |u| \leq |w| = \alpha$ and note that we apply Claim 3.4.5 for the parameter $s' - 1$).

- If α has length p'^{κ} for some $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$, then we choose $b = 1$ if $\alpha = 1^{p'^{\kappa}}$ and $b = 0$ otherwise. Since w satisfies property (3.25), it holds $w \cap \Sigma^{p'^{\kappa}} = \emptyset$. Hence the statements 1, 2, and 3 of Claim 3.4.5 state that the oracle wb is $t_{s'-1}$-valid (again, note that we apply Claim 3.4.5 for the parameter $s' - 1$).

- In all other cases Claim 3.4.5 guarantees that we can choose $b \in \{0, 1\}$ such that wb is $t_{s'-1}$-valid.

By the choice of b, the oracle wb satisfies property (3.25). If wb does not satisfy property (3.24), then by statement 1 of the current claim, $\alpha = c(i', x, F_{i'}^{wb}(x))$ for $i', x \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$, $F_{i'}^{wb}(x) \in \overline{\mathcal{CAN}^{wb}}$, and $\alpha \notin wb$. Claim 3.4.2, however, yields that then $F_{i'}^{w}(x) = F_{i'}^{wb}(x) \in \overline{\mathcal{CAN}^w}$. But then we would have chosen $b = 1$ above, in contradiction to $\alpha \notin wb$.

3. As $w' \sqsubseteq w$ and w satisfies property (3.25), w' satisfies property (3.25). We argue that w' satisfies property (3.24). Let $i', x \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$ such that $|\widetilde{u}| \leq c(i', x, F_{i'}^{w'}(x)) < |w'|$ and $F_{i'}^{w'}(x) \in \overline{\mathcal{CAN}^{w'}}$. As $c(i', x, F_{i'}^{w'}(x)) < |w'|$, Claim 3.4.2 yields $F_{i'}^{w}(x) = F_{i'}^{w'}(x) \in \overline{\mathcal{CAN}^w}$. As w satisfies property (3.24), $c(i', x, F_{i'}^{w'}(x)) \in w$. Since $w' \sqsubseteq w$ and $c(i', x, F_{i'}^{w'}(x)) < |w'|$, we obtain $c(i', x, F_{i'}^{w'}(x)) \in w'$. Hence w' satisfies property (3.24).

This finishes the proof of Claim 3.4.6. □

Note that v is undefined for all words of length $> \gamma(n)$: otherwise, Claim 3.4.6.3 would yield an even smaller $t_{s'-1}$-valid oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies properties (3.24) and (3.25), which contradicts our choice of v. Hence it holds $|v| = |u_z|$ for all $z \in \Sigma^n$.

Claim 3.4.7 *Let $w \in \{v\} \cup \{u_z \mid z \in \Sigma^n\}$.*

1. *For each $\alpha \in w \cap \Sigma^{>n}$ one of the following statements holds.*
   - *$\alpha = c(i', x, F_{i'}^{w}(x))$ for some $i' \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t_s(i', i') \leq \alpha$ and $F_{i'}^{w}(x) \in \overline{\mathcal{CAN}^w}$.*
   - *$\alpha = 1^{p'^{\kappa}}$ for some $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_s)$ and some $\kappa > 0$.*
2. *For all $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_s)$ and all $\kappa > 0$, if $n < p'^{\kappa} \leq \gamma(n)$, then $w \cap \Sigma^{p'^{\kappa}} = \{1^{p'^{\kappa}}\}$.*
3. *For all $z \in \Sigma^n$ and all $\alpha \in u_z - v$ it holds (i) $\alpha = z$ or (ii) $|\alpha| > n$ and $\alpha = c(i', x, F_{i'}^{u_z}(x))$ for some $i' \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t_s(i', i') \leq c(i', x, F_{i'}^{u_z}(x))$ and $F_{i'}^{u_z}(x) \in \overline{\mathcal{CAN}^{u_z}}$.*
4. *For all $z \in \Sigma^n$ and all $\alpha \in v - u_z$ it holds $|\alpha| > n$ and $\alpha = c(i', x, F_{i'}^{v}(x))$ for some $i' \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t_s(i', i') \leq c(i', x, F_{i'}^{v}(x))$ and $F_{i'}^{v}(x) \in \overline{\mathcal{CAN}^{v}}$.*

Proof 1. We first argue for the case $w = u_z$ for some $z \in \Sigma^n$. Let $\alpha \in u_z \cap \Sigma^{>n}$. Moreover, let u' be the prefix of u_z that has length α, i.e., α is the least word that u' is not defined for. In particular, it holds $u' \cap \Sigma^{\leq n} = u_z \cap \Sigma^{\leq n}$ and thus $u' \cap \Sigma^n = \{z\}$. As $u \sqsubseteq u' \sqsubseteq u_z$ and both u and u_z are t_s-valid, Claim 3.4.4 yields that u' is also t_s-valid.

Let us apply Claim 3.4.5 to the oracle u'. If one of the cases 1, 2, 5, and 6 can be applied, then $u'0$ is t_s-valid and can be extended to a t_s-valid oracle u'' with $|u''| = |u_z|$ by Claim 3.4.5. As u'' and u_z agree on all words $< \alpha$ and $\alpha \in u_z - u''$, we obtain $u'' < u_z$ and due to $u' \sqsubseteq u''$ we know that $u'' \cap \Sigma^n = \{z\}$. This is a contradiction to the choice of u_z (recall that u_z is the minimal t_s-valid oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies $u_z \cap \Sigma^n = \{z\}$).

Hence none of the cases 1, 2, 5, and 6 of Claim 3.4.5 can be applied, i.e., either (i) Claim 3.4.5.3 or (ii) Claim 3.4.5.4 can be applied. Hence either (i) $\alpha = 1^{p'^\kappa}$ for some $p' \in \mathbb{P}_{\geq 3}$ and $\kappa > 0$ with $-p' \in \mathrm{ran}(t_s)$ or (ii) $\alpha = c(i', x, F_{i'}^{u_z}(x))$ for $i' \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t_s(i', i') \leq \alpha$. In the latter case, as $\alpha \in u_z$ and u_z is t_s-valid, we obtain from V1 that $F_{i'}^{u_z}(x) \in \overline{\mathcal{CAN}^{u_z}}$.

The arguments for the case $w = v$ are similar, yet a little more complicated. Let $\alpha \in v \cap \Sigma^{>n}$. Moreover, let v' be the prefix of v that has length α, i.e., α is the least word that v' is not defined for. As $\widetilde{u} \sqsubseteq v' \sqsubseteq v$ and both $\widetilde{u}$ and v are $t_{s'-1}$-valid, Claim 3.4.4 yields that v' is also $t_{s'-1}$-valid. Moreover, by Claim 3.4.6.3, v' satisfies properties (3.24) and (3.25).

Let us apply Claim 3.4.5 to the oracle v' (for the parameter $s' - 1$). If one of the cases 1, 2, 5, and 6 can be applied, then $v'0$ is $t_{s'-1}$-valid.
First, assume that it does not hold that $v'0$ satisfies properties (3.24) and (3.25). If $v'0$ does not satisfy property (3.25), then $\alpha = 1^{p'^\kappa}$ for some $p' \in \mathbb{P}_{\geq 3}$ with $-p' \in \mathrm{ran}(t_s)$ and $\kappa > 0$. If $v'0$ does not satisfy property (3.24), then it holds by Claim 3.4.6.1 that $\alpha = c(i', x, y)$ for some $i', x, y \in \mathbb{N}$ with $i' > 0$ and $t_s(i', i') > 0$. As v is $t_{s'-1}$-valid and $\alpha \in v$, it holds $F_{i'}^v(x) = y \in \overline{\mathcal{CAN}^v}$. Moreover, by construction, $\alpha > |u| \geq t_s(i', i')$. Hence under the assumption that $v'0$ does not satisfy property (3.24) or $v'0$ does not satisfy property (3.25), we obtain that α is of the form described in statement 1 of the current claim.
Now we consider the case that $v'0$ satisfies properties (3.24) and (3.25) and show that this assumption leads to a contradiction. By iteratively applying Claim 3.4.6.2 we extend $v'0$ to a $t_{s'-1}$-valid oracle v'' that satisfies $|v''| = |v|$ and properties (3.24) and (3.25). As v'' and v agree on all words $< \alpha$, $\alpha \in v - v''$, and $|v| = |v''|$, it holds $v'' < v$, in contradiction to the choice of v (recall that v is the minimal $t_{s'-1}$-valid oracle $\sqsupsetneq \widetilde{u}$ that is defined for all words of length $\leq \gamma(n)$ and that satisfies properties (3.24) and (3.25)).

In order to finish the proof of statement 1, it remains to consider the cases that Claim 3.4.5.3 or Claim 3.4.5.4 can be applied to v'. This means that either (i) there exist $p' \in \mathbb{P}_{\geq 3}$ and $\kappa \in \mathbb{N}^+$ with $-p' \in \mathrm{ran}(t_{s'-1}) \subseteq \mathrm{ran}(t_s)$ such that $\alpha = 1^{p'^\kappa}$, or (ii) $\alpha = c(i', x, y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_{s'-1}(i', i') = t_s(i', i') \leq \alpha$. In the latter case, as $\alpha \in v$ and v is $t_{s'-1}$-valid, we obtain from V1 that $F_{i'}^v(x) = y \in \overline{\mathcal{CAN}^v}$. This shows that also in this case α is of the form described in statement 1 of the current claim.

2. The statement is true in case $w = v$ as v satisfies property (3.25). Let us argue for the case $w = u_z$ for some $z \in \Sigma^n$. Let p' and κ have the properties mentioned in statement 2. As $-p' \in \mathrm{ran}(t_s)$, u_z is t_s-valid, and u_z is defined for all words of length p'^κ, V3 yields that there exists $\beta \in \Sigma^{p'^\kappa} \cap u_z$. Let β be the minimal element of $\Sigma^{p'^\kappa} \cap u_z$. It suffices to show $\beta = 1^{p'^\kappa}$. For a contradiction, we assume $\beta < 1^{p'^\kappa}$. Let u' be the prefix of u_z that is defined for exactly the words $< \beta$. Then $u \sqsubseteq u' \sqsubseteq u_z$ and both u and u_z are t_s-valid. Hence by Claim 3.4.4, the oracle u' is t_s-valid as well.

By Claim 3.4.5, u' can be extended to a t_s-valid oracle u'' that satisfies $|u''| = |u_z|$ and $u'' \cap \Sigma^{p'^\kappa} = \{1^{p'^\kappa}\}$ (the latter property can be achieved by iteratively applying Claim 3.4.5.2 and then applying Claim 3.4.5.3 once when extending the respective oracle for the words of length

p'^{κ}). Then $\beta \in u_z - u''$. As the oracles u'' and u_z agree on all words $< \beta$ and it holds $|u''| = |u_z|$, we have $u'' < u_z$ and $u'' \cap \Sigma^n = \{z\}$, in contradiction to the choice of u_z (again, recall that u_z is the minimal t_s-valid oracle that is defined for all words of length $\leq \gamma(n)$ and that satisfies $u_z \cap \Sigma^n = \{z\}$).

3. Recall that u is defined for exactly the words of length $< n$. By $u_z \sqsupseteq u$, $v \sqsupseteq \widetilde{u} \sqsupseteq u$, and $u_z \cap \Sigma^n = \{z\}$, either $\alpha = z$ or $|\alpha| > n$. Thus statements 1 and 2 of the present claim complete the proof of statement 3.

4. The fact that $u_z \sqsupseteq u$, $v \sqsupseteq \widetilde{u} \sqsupseteq u$, and $\widetilde{u} \cap \Sigma^n = \emptyset$ yields $|\alpha| > n$. Thus statements 1 and 2 of the present claim complete the proof of statement 4.

This finishes the proof of Claim 3.4.7. □

Let us study the case that both computations $M_i^v(F_r^v(0^n))$ and $M_j^v(F_r^v(0^n))$ reject. Then they even definitely reject as v is defined for all words of length $\leq \gamma(n)$. For the same reason the computation $F_r^v(0^n)$ is definite. But then v is not only $t_{s'-1}$-valid but also t-valid for $t = t_{s'-1} \cup \{(i,j) \mapsto 0\}$ and the construction would have chosen $t_{s'} = t$, in contradiction to $t_s(i,j) = -p < 0$. Hence one of the computations $M_i^v(F_r^v(0^n))$ and $M_j^v(F_r^v(0^n))$ accepts and by the (sufficiently long) choice of v even definitely accepts. By symmetry, it suffices to consider the case that $M_i^v(F_r^v(0^n))$ definitely accepts.

Let U be the set that consists of all oracle queries of $F_r^v(0^n)$ and all oracle queries of the least accepting path of $M_i^v(F_r^v(0^n))$. Observe $\ell(U) \leq \gamma(n)$. Moreover, define $Q_0(U) = U$ and for $m \in \mathbb{N}$,

$$Q_{m+1}(U) = \bigcup_{\substack{i',x,y \in \mathbb{N}, i' > 0 \\ c(i',x,y) \in Q_m(U)}} \Big[\{q \mid q \text{ is queried by } F_{i'}^v(x)\} \cup \{q \mid y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x' \rangle \text{ for some } i'' > 0 \text{ and } x' \in \mathbb{N}, M_{i''}^v(x') \text{ has an accepting path, and } q \text{ is queried on the least such path}\} \Big].$$

Let $Q(U) = \bigcup_{m \in \mathbb{N}} Q_m(U)$.

Claim 3.4.8 *$\ell(Q(U)) \leq 2\gamma(n)$ and the length of each word in $Q(U)$ is $\leq \gamma(n)$.*

Proof We show that for all $m \in \mathbb{N}$, $\ell(Q_{m+1}(U)) \leq 1/2 \cdot \ell(Q_m(U))$. Then $\sum_{m=0}^{\kappa} 1/2^m \leq 2$ for all $\kappa \in \mathbb{N}$ implies $\ell(Q(U)) \leq 2 \cdot \ell(U) \leq 2\gamma(n)$. Moreover, from $\ell(U) \leq \gamma(n)$ and $\ell(Q_{m+1}(U)) \leq 1/2 \cdot \ell(Q_m(U))$ the second part of the claim follows.

Let $m \in \mathbb{N}$ and consider an arbitrary element α of $Q_m(U)$. If α is not of the form $c(i',x,y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$, then α generates no elements in $Q_{m+1}(U)$. Assume $\alpha = c(i',x,y)$ for $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$. The computation $F_{i'}^v(x)$ runs for at most $|x|^{i'} + i' < |\alpha|/4$ steps, where "$<$" holds by (3.18). Hence the set of queries Q of $F_{i'}^v(x)$ satisfies $\ell(Q) \leq |\alpha|/4$.

If y is not of the form $\langle 0^{i''}, 0^{|x'|^{i''}+i''}, x' \rangle$ for $i'' \in \mathbb{N}^+$ and $x' \in \mathbb{N}$, then α does not generate any further elements in $Q_{m+1}(U)$ than the queries asked by $F_{i'}^v(x)$. Let us assume $y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x' \rangle$ for some $i'' \in \mathbb{N}^+$ and some $x' \in \mathbb{N}$. Then the computation $M_{i''}^v(x)$ runs for less than $|y| < |\alpha|/4$ steps, where again "$<$" holds by (3.18). Hence, for the set Q of queries of the least accepting path of the computation $M_{i''}^v(x)$ (if such a path exists) we have $\ell(Q) \leq |\alpha|/4$.

Consequently,

$$\ell(Q_{m+1}(U)) \leq \sum_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q_m(U)}} \Big[\underbrace{\ell\big(\{q \mid q \text{ is queried by } F^v_{i'}(x)\}\big)}_{\leq |c(i',x,y)|/4} + \underbrace{\ell\big(\{q \mid y = \langle 0^{i''}, 0^{|x'|^{i''}+i''}, x'\rangle \text{ for some } i''>0 \text{ and } x'\in\mathbb{N},\ M^v_{i''}(x') \text{ has an accepting path, and } q \text{ is queried on the least such path}\}\big)\Big]}_{\leq |c(i',x,y)|/4}$$

$$\leq \sum_{\substack{i',x,y\in\mathbb{N},i'>0\\ c(i',x,y)\in Q_m(U)}} |c(i',x,y)|/2 \leq \ell(Q_m(U))/2,$$

which finishes the proof of Claim 3.4.8. □

For z even we say that u_z and v conflict if there exists $\alpha \in Q(U)$ with $\alpha \in u_z \triangle v$. In that case we say that u_z and v conflict in α. As $u_z \sqsupseteq u$ and $v \sqsupseteq \widetilde{u} \sqsupseteq u$, either u_z and v conflict in a word of length $\geq n$ or they do not conflict at all.

Claim 3.4.9 *For all even $z \in \Sigma^n$, if v and u_z conflict, then they conflict in z.*

Proof Let $z \in \Sigma^n$ be even such that u_z and v conflict. By Claim 3.4.7.3 and Claim 3.4.7.4, it is sufficient to show that v and u_z conflict in a word of length n.

As v and u_z are both extensions of u, u_z and v do not conflict in a word of length $< n$. Let $\alpha \in Q(U)$ be the least word of length $> n$ that u_z and v conflict in (if such a word does not exist, then u_z and v only conflict in words of length n). As $\alpha \in v \triangle u_z$, it suffices to study the following two cases.

- Assume $\alpha \in u_z - v$. By Claim 3.4.7.3, it holds $\alpha = c(i',x,y)$ for some $i' \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ with $0 < t_s(i',i') \leq c(i',x,F^{u_z}_{i'}(x))$ and $F^{u_z}_{i'}(x) = y \in \overline{\mathcal{CAN}^{u_z}}$.

  First assume $F^v_{i'}(x) \neq y$. Then there is some query q of $F^v_{i'}(x)$ that is in $v \triangle u_z$ (otherwise, $F^{u_z}_{i'}(x)$ and $F^v_{i'}(x)$ would output the same value). As v and u_z agree on all words of length $< n$, it holds $|q| \geq n$. Since $|q| \leq |x|^{i'} + i' < |c(i',x,y)| = |\alpha|$ and α is the least word of length $> n$ that v and u_z conflict in, it holds $|q| = n$. By $\alpha \in Q(U)$ and the definition of $Q(U)$, it holds $q \in Q(U)$. Hence v and u_z conflict in a word of length n.

  Now assume $F^v_{i'}(x) = y$. As $\alpha \notin v$ and v satisfies property (3.24), it holds $y \notin \overline{\mathcal{CAN}^v}$. As $y \in \mathcal{CAN}^v$, y is of the form $\langle 0^{i''}, 0^{|x'|^{i''}}, x'\rangle$ for some $i'' > 0$ and $x' \in \mathbb{N}$. It follows from $y \in \mathcal{CAN}^v$ that the computation $M^v_{i''}(x')$ has an accepting path. By the definition of $Q(U)$, all queries q that are asked on the least such path are in $Q(U)$. However, $y \in \overline{\mathcal{CAN}^{u_z}}$ yields that there is some query q on the least accepting path of $M^v_{i''}(x')$ that is in $v \triangle u_z$ (otherwise, $M^{u_z}_{i''}(x')$ would accept as well). As v and u_z agree on all words of length $< n$, it holds $|q| \geq n$. Since $|q| \leq |x'|^{i''} + i'' < |y| < |c(i',x,y)| = |\alpha|$ and α is the least word of length $> n$ that v and u_z conflict in, it holds $|q| = n$. Hence v and u_z conflict in a word of length n.

- Assume $\alpha \in v - u_z$. By Claim 3.4.7.4, it holds $\alpha = c(i',x,F^v_{i'}(x))$ for some $i' \in \mathbb{N}^+$ and $x \in \mathbb{N}$ with $0 < t_s(i',i') \leq c(i',x,F^v_{i'}(x))$ and $F^v_{i'}(x) \in \overline{\mathcal{CAN}^v}$. If $F^{u_z}_{i'}(x) = F^v_{i'}(x)$, then by V5, we have $\alpha \in u_z$, a contradiction. Hence $F^{u_z}_{i'}(x) \neq F^v_{i'}(x)$. Then there is some query q of $F^v_{i'}(x)$ that is in $v \triangle u_z$ (otherwise, $F^{u_z}_{i'}(x)$ and $F^v_{i'}(x)$ would output the

same value). As v and u_z agree on all words of length $< n$, it holds $|q| \geq n$. Since $|q| \leq |x|^{i'} + i' < |c(i', x, F^v_{i'}(x))| = |\alpha|$ and α is the least word of length $> n$ that v and u_z conflict in, it holds $|q| = n$. By $\alpha \in Q(U)$ and the definition of $Q(U)$, it holds $q \in Q(U)$. Hence v and u_z conflict in a word of length n.

In both cases v and u_z conflict in a word of length n. This finishes the proof of Claim 3.4.9. □

By Claim 3.4.8 and (3.23), $|Q(U)| \leq \ell(Q(U)) \leq 2\gamma(n) < 2^{n-1} = |\{z \in \Sigma^n \mid z \text{ even}\}|$. Hence there exists an even $z \in \Sigma^n$ with $z \notin Q(U)$. Consequently, v and u_z do not conflict in z. Then by Claim 3.4.9, v and u_z do not conflict at all.

As all queries of $F^v_r(0^n)$ and all queries of the least accepting path of $M^v_i(F^v_r(0^n))$ are in $U \subseteq Q(U)$, v and u_z agree on all these queries, and u_z is defined for all words of length $\leq \gamma(n)$, it holds that $F^{u_z}_r(0^n) = F^v_r(0^n)$ is definite and that $M^{u_z}_i(F^{u_z}_r(0^n))$ definitely accepts. Note that since u_z is defined for all words of length $\leq \gamma(n)$, the computation $M^{u_z}_j(F^{u_z}_r(0^n))$ is definite as well. We study two cases depending on whether this computation accepts or rejects.

- First consider the case that $M^{u_z}_j(F^{u_z}_r(0^n))$ definitely rejects. As z is even, $0^n \in B^{u_z}_p$ and clearly $0^n \in B^w_p$ for all $w \sqsupseteq u_z$. This, however, contradicts the assumption that step s of the construction treating the task (i, j, r) is not possible.
- Next we consider the case that $M^{u_z}_j(F^{u_z}_r(0^n))$ definitely accepts. Then both $M^{u_z}_i(F^{u_z}_r(0^n))$ and $M^{u_z}_j(F^{u_z}_r(0^n))$ definitely accept. As u_z is $t_{s'-1}$-valid by Claim 3.4.3, we obtain that u_z is even t-valid for $t = t_{s'-1} \cup \{(i, j) \mapsto 0\}$. But then the construction would have chosen $t_{s'} = t$, in contradiction to $t_s(i, j) = -p < 0$.

As in both cases we obtain a contradiction, the construction described above is possible. It remains to show that relative to the final oracle O, there exist P^O-optimal proof systems for $\overline{\mathcal{CAN}^O}$ and $\mathrm{NP}^O \cap \mathrm{coNP}^O$ does not contain problems that are $\leq^{\mathrm{p},O}_{\mathrm{m}}$-hard for $\mathrm{UP}^O \cap \mathrm{coUP}^O$.

Claim 3.4.10 *$\overline{\mathcal{CAN}^O}$ has P^O-optimal proof systems.*

Proof Let $g \in \mathrm{FP}^O$ be an arbitrary proof system for $\overline{\mathcal{CAN}^O}$ and a be an arbitrary element of $\overline{\mathcal{CAN}^O}$. Define f to be the following function $\Sigma^* \to \Sigma^*$:

$$f(z) = \begin{cases} g(z') & \text{if } z = 1z' \\ y & \text{if } z = 0c(i, x, y) \text{ for } i \in \mathbb{N}^+,\ x, y \in \mathbb{N}, \text{ and } c(i, x, y) \in O \\ a & \text{otherwise.} \end{cases}$$

By definition, $f \in \mathrm{FP}^O$ and as g is a proof system for $\overline{\mathcal{CAN}^O}$ it holds $f(\Sigma^*) \supseteq \overline{\mathcal{CAN}^O}$. We show $f(\Sigma^*) \subseteq \overline{\mathcal{CAN}^O}$. Let $z \in \Sigma^*$. Assume $z = 0c(i, x, y)$ for $i \in \mathbb{N}^+$, $x, y \in \mathbb{N}$, and $c(i, x, y) \in O$ (otherwise, clearly $f(z) \in \overline{\mathcal{CAN}^O}$). For each large enough s it holds that w_s is defined for $c(i, x, y)$, i.e. $w_s(c(i, x, y)) = 1$, and as w_s is t_s-valid, we obtain by V1 that $f(z) = y \in \overline{\mathcal{CAN}^{w_s}}$ and by Claim 3.4.2 that $y \in \overline{\mathcal{CAN}^v}$ for all $v \sqsupseteq w_s$. This shows that f is a proof system for $\overline{\mathcal{CAN}^O}$.

It remains to show that each proof system for $\overline{\mathcal{CAN}^O}$ is P^O-simulated by f. Let h be an arbitrary proof system for $\overline{\mathcal{CAN}^O}$. Then there exists $i > 0$ such that F^O_i computes h. By construction, $t_s(i, i) > 0$ for s being the number of the step that treats the task (i, i). Consider the following function $\pi\colon \Sigma^* \to \Sigma^*$:

$$\pi(x) = \begin{cases} 0c(i, x, F^O_i(x)) & \text{if } c(i, x, F^O_i(x)) \geq t_s(i, i) \\ z & \text{if } c(i, x, F^O_i(x)) < t_s(i, i) \text{ and } z \text{ is minimal with } f(z) = F^O_i(x) \end{cases}$$

As f and F_i^O are proof systems for $\overline{\mathcal{CAN}^O}$, for every x there exists z with $f(z) = F_i^O(x)$. Hence π is total. Since $t_s(i,i)$ is a constant, $\pi \in \mathrm{FP}^O$. It remains to show that $f(\pi(x)) = F_i^O(x)$ for all $x \in \Sigma^*$. If $c(i,x,F_i^O(x)) < t_s(i,i)$, this holds. Otherwise, choose s' large enough such that (i) $s' \geq s$ (i.e., $t_{s'}(i,i) = t_s(i,i) > 0$) and (ii) $w_{s'}$ is defined for $c(i,x,F_i^{w_{s'}}(x))$. Then, as $w_{s'}$ is $t_{s'}$-valid, V5 yields that $c(i,x,F_i^{w_{s'}}(x)) \in w_{s'}$. By Claim 3.4.2, $F_i^{w_{s'}}(x)$ is definite and hence $F_i^O(x) = F_i^{w_{s'}}(x)$ as well as $c(i,x,F_i^O(x)) \in w_{s'} \subseteq O$. Hence $f(\pi(x)) = f(0c(i,x,F_i^O(x))) = F_i^O(x)$. □

Claim 3.4.11 $\mathrm{NP}^O \cap \mathrm{coNP}^O$ *does not contain problems that are* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-hard for* $\mathrm{UP}^O \cap \mathrm{coUP}^O$.

Proof Assume the assertion is wrong, i.e., there exist distinct $i,j \in \mathbb{N}^+$ such that $L(M_i^O), L(M_j^O) \in \mathrm{NP}^O$ with $L(M_i^O) = \overline{L(M_j^O)}$ and for every $A \in \mathrm{UP}^O \cap \mathrm{coUP}^O$ it holds $A \leq_{\mathrm{m}}^{\mathrm{p},O} L(M_i^O)$. From $L(M_i^O) = \overline{L(M_j^O)}$ if follows that for all s there does not exist z such that (i) both $M_i^{w_s}(z)$ and $M_j^{w_s}(z)$ definitely accept or (ii) both $M_i^{w_s}(z)$ and $M_j^{w_s}(z)$ definitely reject. Hence by V2, there does not exist s with $t_s(i,j) = 0$ and thus by construction, $t_s(i,j) = -p$ for some $p \in \mathbb{P}_{\geq 3}$ and all sufficiently large s. The latter implies $|O \cap \Sigma^{p^k}| = 1$ for all $k > 0$ (cf. V3), which yields $A_p^O = \overline{B_p^O}$ and $A_p^O, B_p^O \in \mathrm{UP}^O$, i.e., $A_p^O \in \mathrm{UP}^O \cap \mathrm{coUP}^O$. Thus there exists r such that $A_p^O \leq_{\mathrm{m}}^{\mathrm{p},O} L(M_i^O)$ via F_r^O. Let s be the step that treats task (i,j,r). This step makes sure that there exists $n \in \mathbb{N}^+$ such that at least one of the following properties holds:

- $\forall_{\widetilde{w} \sqsupseteq w_s}\, 0^n \in A_p^{\widetilde{w}}$, $F_r^{w_s}(0^n)$ is definite, and $M_i^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.
- $\forall_{\widetilde{w} \sqsupseteq w_s}\, 0^n \in B_p^{\widetilde{w}}$, $F_r^{w_s}(0^n)$ is definite, and $M_j^{w_s}(F_r^{w_s}(0^n))$ definitely rejects.

As $O(q) = w_s(q)$ for all q that w_s is defined for, one of the following two statements holds.

- $0^n \in A_p^O$ and $F_r^O(0^n) \notin L(M_i^O)$.
- $0^n \in B_p^O = \overline{A_p^O}$ and $F_r^O(0^n) \notin L(M_j^O) = \overline{L(M_i^O)}$.

This is a contradiction to $A_p^O \leq_{\mathrm{m}}^{\mathrm{p},O} L(M_i^O)$ via F_r^O, which completes the proof of Claim 3.4.11. □

This finishes the proof of Theorem 3.4.1. □

Corollary 3.4.12 *There exists an oracle* O *with the following properties:*

1. $\mathrm{NP}^O \cap \mathrm{coNP}^O$ *does not contain* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problems.*
2. $\mathrm{UP}^O \cap \mathrm{coUP}^O$ *does not contain* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete problems.*
3. *Each non-empty set in* coNP^O *has* P^O*-optimal proof systems.*
4. *There do not exist problems that are* $\leq_{\mathrm{m}}^{\mathrm{p},O}$*-complete for* NP^O *and have* P^O*-optimal proof systems.*

Proof Statements 1, 2, and 3 follow from Theorem 3.4.1 and Corollary 2.3.6. Köbler, Messner, and Torán [KMT03, Theorem 4.1] show that if both NP and coNP contain $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets that have P-optimal proof systems, then NP ∩ coNP has $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets. Their proof is relativizable, i.e., if both coNP^O and NP^O contain $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete sets that have P^O-optimal proof systems, then $\mathrm{NP}^O \cap \mathrm{coNP}^O$ has $\leq_{\mathrm{m}}^{\mathrm{p},O}$-complete elements. Thus statement 3 and the negation of statement 4 imply that statement 1 does not hold. This proves statement 4. □

3.5 P ≠ NP, ¬CON, and ¬SAT Relative to an Oracle

In this section we construct an oracle relative to which P ≠ NP and there exist P-optimal proof systems for both all non-empty sets in NP and all non-empty sets in coNP, which separates the (relativized) conjectures P ≠ NP and $\mathsf{CON} \vee \mathsf{SAT}$. This oracle construction is the least complex among all oracles we construct in this thesis.

Theorem 3.5.1 *There exists an oracle O relative to which the following statements hold:*

- $\mathrm{P}^O \neq \mathrm{NP}^O$
- $\mathcal{CAN}^O$ *has* P^O*-optimal proof systems.*
- $\overline{\mathcal{CAN}^O}$ *has* P^O*-optimal proof systems.*

Proof We define $c(i,x,y) = \langle 0^i, 0^{|x|^i+i}, x, y\rangle$. Let D be a (possibly partial) oracle and define

$$A^D \quad = \quad \{0^n \mid \exists_{y\in\Sigma^n} 0y \in D\}.$$

We will construct the oracle such that $A^O \in \mathrm{NP}^O - \mathrm{P}^O$ for the final oracle O. Note that throughout this proof we sometimes omit the oracles in the superscript, e.g., we write NP or A instead of NP^D or A^D. However, we do not do that in the "actual" proof but only when explaining ideas in a loose way in order to give the reader some intuition.

Let us briefly sketch the idea of our construction.

Preview of the Construction For each F_i we first try to ensure that F_i does not compute a proof system for $\mathcal{CAN}$ (resp., $\overline{\mathcal{CAN}}$). If this is impossible, then F_i inherently computes a proof system for $\mathcal{CAN}$ (resp., $\overline{\mathcal{CAN}}$). In that case we start to encode the values of F_i into the oracle so that F_i can be P-simulated by some proof system for $\mathcal{CAN}$ (resp., $\overline{\mathcal{CAN}}$) that we will define later and finally show to be P-optimal.

Moreover, we diagonalize against all P_i so that A is not in P relative to the final oracle.

The following claim can be proven in the same way as Claim 3.3.4. The only difference is that we use a slightly different coding function c here.

Claim 3.5.2 ([DG19]) *Let $w \in \Sigma^*$, $i \in \mathbb{N}^+$, and $x, y \in \mathbb{N}$ such that $c(i,x,y) \leq |w|$. Then the following holds.*

1. *$F_i^w(x)$ is definite and $F_i^w(x) < |w|$.*
2. *For all $v \sqsupseteq w$, $\big(F_i^w(x) \in \mathcal{CAN}^w \Leftrightarrow F_i^w(x) \in \mathcal{CAN}^v\big)$.*

During the construction we maintain a collection of requirements, which is represented by a function in $\mathcal{T} := \{t\colon \{0,1\} \times \mathbb{N}^+ \to \mathbb{N} \mid t \text{ has a finite domain}\}$. Let $t \in \mathcal{T}$. A partial oracle $w \in \Sigma^*$ is called t-valid if it satisfies the following properties.

V1 For all $i \in \mathbb{N}^+$,

1. if $10c(i,x,y) \in w$ for some $x, y \in \mathbb{N}$, then $F_i^w(x) = y \in \mathcal{CAN}^w$.
2. if $11c(i,x,y) \in w$ for some $x, y \in \mathbb{N}$, then $F_i^w(x) = y \in \overline{\mathcal{CAN}^w}$.

(meaning: if the oracle contains the codeword $1bc(i,x,y)$ for $b = 0$ (resp., $b = 1$), then $F_i^w(x)$ outputs y and $y \in \mathcal{CAN}^w$ (resp., $y \in \overline{\mathcal{CAN}^w}$); hence $1bc(i,x,y) \in w$ is a proof for $y \in \mathcal{CAN}^w$ (resp., $y \in \overline{\mathcal{CAN}^w}$).)

V2 For all $i \in \mathbb{N}^+$, if $t(0,i) = 0$, then there exists x such that $F_i^w(x)$ is definite and $F_i^w(x) \notin \mathcal{CAN}^v$ for all $v \sqsupseteq w$.
(meaning: for every extension of the current oracle, F_i is not a proof system for $\mathcal{CAN}$.)

V3 For all $i \in \mathbb{N}^+$, if $t(0,i) > 0$, then for all $x \in \mathbb{N}$ with $t(0,i) \leq 10c(i,x,F_i^w(x)) < |w|$, it holds $10c(i,x,F_i^w(x)) \in w$.
(meaning: if $t(0,i) > 0$, then from $t(0,i)$ on, we encode the values of F_i into the oracle.)

V4 For all $i \in \mathbb{N}^+$, if $t(1,i) = 0$, then there exists x such that $F_i^w(x)$ is definite and $F_i^w(x) \notin \overline{\mathcal{CAN}}^v$ for all $v \sqsupseteq w$.
(meaning: for every extension of the current oracle, F_i is not a proof system for $\overline{\mathcal{CAN}}$.)

V5 For all $i \in \mathbb{N}^+$, if $t(1,i) > 0$, then for all $x \in \mathbb{N}$ with $t(1,i) \leq 11c(i,x,F_i^w(x)) < |w|$, it holds $11c(i,x,F_i^w(x)) \in w$.
(meaning: if $t(1,i) > 0$, then from $t(1,i)$ on, we encode the values of F_i into the oracle.)

This definition directly implies the following claim.

Claim 3.5.3 *Let $t, t' \in \mathcal{T}$ such that t' is an extension of t. If $w \in \Sigma^*$ is t'-valid, then w is t-valid.*

Claim 3.5.4 *Let $t \in \mathcal{T}$ and $u, v, w \in \Sigma^*$ so that $u \sqsubseteq v \sqsubseteq w$. If u and w are t-valid, then v is t-valid.*

Proof The oracle v satisfies V2 and V4, since u satisfies V2 and V4 and $u \sqsubseteq v$.

Let us argue for V1. Let $10c(i,x,y) \in v$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$. Then $10c(i,x,y) \in w$ and as w is t-valid, it holds by V1 that $F_i^w(x) = y \in \mathcal{CAN}^w$. By Claim 3.5.2, $F_i^v(x) = F_i^w(x) = y$ and $\mathcal{CAN}^v(y) = \mathcal{CAN}^w(y) = 1$. Analogously, $11c(i,x,y) \in v$ for $i \in \mathbb{N}^+$ and $x, y \in \mathbb{N}$ implies $F_i^v(x) = y \in \overline{\mathcal{CAN}}^v$. Thus v satisfies V1.

It remains to prove that v satisfies V3 and V5. We argue for these properties in parallel. Let $i \in \mathbb{N}^+$, $x \in \mathbb{N}$, and $b \in \{0,1\}$ such that $0 < t(b,i) \leq 1bc(i,x,F_i^v(x)) < |v|$. Then by Claim 3.5.2, $F_i^v(x) = F_i^w(x)$. As w is t-valid, we obtain by V3 or by V5 that $1bc(i,x,F_i^w(x)) \in w$. Since $1bc(i,x,F_i^v(x)) < |v|$ and $v \sqsubseteq w$, we have $1bc(i,x,F_i^v(x)) = 1bc(i,x,F_i^w(x)) \in v$, which shows that v satisfies V3 and V5. □

Oracle construction. Let $T\colon \mathbb{N} \to \{0,1,2\} \times \mathbb{N}^+$ be a bijection. Each value $T(s)$ for some $s \in \mathbb{N}$ stands for a task. We treat the tasks in the order specified by T. We start with the unique nowhere defined function t_0 in $\mathcal{T}$ and the t_0-valid oracle $w_0 = \varepsilon$. Then we define functions $t_1, t_2, \ldots$ in $\mathcal{T}$ such that t_{i+1} is an extension of t_i and partial oracles $w_0 \sqsubsetneq w_1 \sqsubsetneq w_2 \sqsubsetneq \ldots$ such that each w_i is t_i-valid. Finally, we choose $O = \bigcup_{i=0}^{\infty} w_i$ (note that O is totally defined, since in each step we strictly extend the oracle). We describe step $s > 0$, which starts with a t_{s-1}-valid oracle w_{s-1} and extends it to a t_s-valid $w_s \sqsupsetneq w_{s-1}$ depending on $T(s)$. We will argue later that the construction is possible.

- task $(0,i)$ for $i \in \mathbb{N}^+$: Let $t' = t_{s-1} \cup \{(0,i) \mapsto 0\}$. If there exists some t'-valid $v \sqsupsetneq w_{s-1}$, then let $t_s = t'$ and define $w_s = v$ for the least t'-valid $v \sqsupsetneq w_{s-1}$. Otherwise, let $t_s = t_{s-1} \cup \{(0,i) \mapsto \max(1, |w_{s-1}|)\}$ and choose $b \in \{0,1\}$ and $w_s = w_{s-1}b$ such that w_s is t_s-valid.

- task $(1,i)$ for $i \in \mathbb{N}^+$: Let $t' = t_{s-1} \cup \{(1,i) \mapsto 0\}$. If there exists some t'-valid $v \sqsupsetneq w_{s-1}$, then let $t_s = t'$ and define $w_s = v$ for the least t'-valid $v \sqsupsetneq w_{s-1}$. Otherwise, let $t_s = t_{s-1} \cup \{(1,i) \mapsto \max(1, |w_{s-1}|)\}$ and choose $b \in \{0,1\}$ and $w_s = w_{s-1}b$ such that w_s is t_s-valid.

- task $(2, i)$ for $i \in \mathbb{N}^+$: Let $t_s = t_{s-1}$ and chose $w_s \supsetneq w_{s-1}$ such that there exists some $n \in \mathbb{N}$ such that w_s is defined for all words of length $n^i + i$ and the following equivalence holds:
$$0^n \in A^{w_s} \Leftrightarrow P_i^{w_s}(0^n) \text{ rejects.}$$
(meaning: we ensure that P_i does not accept A relative to the final oracle.)

Claim 3.5.5 *Let $s \geq 0$, $w \sqsupseteq w_s$ be t_s-valid, and $z = |w|$.*

1. *If $z = 10c(i, x, F_i^w(x))$ for some $i, x \in \mathbb{N}$ with $i > 0$ and $0 < t_s(0, i) \leq z$, then $w1$ is t_s-valid.*
2. *If $z = 11c(i, x, F_i^w(x))$ for some $i, x \in \mathbb{N}$ with $i > 0$ and $0 < t_s(1, i) \leq z$, then $w1$ is t_s-valid.*
3. *If there exist $n \in \mathbb{N}$ and $y \in \Sigma^n$ such that $z = 0y$, then $w0$ and $w1$ are t_s-valid.*
4. *In all other cases (i.e., none of the assumptions in 1–3 holds) $w0$ is t_s-valid.*

Proof First observe that V2 and V4 are not affected by extending the oracle. Moreover, by Claim 3.5.2 and as w satisfies V1, V3, and V5, it holds that the oracle wb for $b \in \{0, 1\}$ satisfies

(A) V1.1 if $\left[b = 1 \wedge \exists_{i,x,y \in \mathbb{N}, i>0}\, z = 10c(i, x, y) \wedge \neg(F_i^w(x) = y \in \mathcal{CAN}^w)\right]$ does not hold.

(B) V1.2 if $\left[b = 1 \wedge \exists_{i,x,y \in \mathbb{N}, i>0}\, z = 11c(i, x, y) \wedge \neg(F_i^w(x) = y \in \overline{\mathcal{CAN}^w})\right]$ does not hold.

(C) V3 if $\left[b = 0 \wedge \exists_{i,x \in \mathbb{N}, i>0}\, z = 10c(i, x, F_i^w(x)) \wedge 0 < t_s(0, i) \leq z\right]$ does not hold.

(D) V5 if $\left[b = 0 \wedge \exists_{i,x \in \mathbb{N}, i>0}\, z = 11c(i, x, F_i^w(x)) \wedge 0 < t_s(1, i) \leq z\right]$ does not hold.

This proves statement 3. Let us argue for statement 4. According to (A) and (B) $w0$ satisfies V1. If $w0$ does not satisfy V3 (resp., V5), then according to (C) (resp., (D)) it holds $z = 10c(i, x, F_i^w(x))$ (resp., $z = 11c(i, x, F_i^w(x))$) for some $i, x \in \mathbb{N}$ with $i > 0$ and $0 < t_s(0, i) \leq z$ (resp., $0 < t_s(1, i) \leq z$). However, this case is covered by statement 1 (resp., statement 2), which completes the proof of statement 4.

Let us consider statement 1. By (C) and (D), it suffices to argue for V1. By (B), the oracle $w1$ satisfies V1.2. By (A), it is sufficient to show $F_i^w(x) \in \mathcal{CAN}^w$. For a contradiction assume $F_i^w(x) \notin \mathcal{CAN}^w$. Let $s' > 0$ be the step with $T(s') = (0, i)$. Then $s' \leq s$ as $t_s(0, i)$ is defined. By Claim 3.5.3, the oracle w is $t_{s'-1}$-valid and by Claim 3.5.2, $F_i^w(x)$ is definite and $F_i^w(x) \notin \mathcal{CAN}^v$ for all $v \sqsupseteq w$. Hence w is even t-valid for $t = t_{s'-1} \cup \{(0, i) \mapsto 0\}$. But then the construction would have chosen $t_{s'} = t$, in contradiction to $t_s(0, i) > 0$.

Statement 2 can be proven analogously. □

We now show that the described construction is possible: for a contradiction, assume that it is not. Hence there exists a minimal $s > 0$ such that step s fails. Note that then the construction still defines a t_{s-1}-valid partial oracle $w_{s-1} \in \Sigma^*$.

Assume that in step s some task (a, i) for $a \in \{0, 1\}$ and $i \in \mathbb{N}^+$ is treated. Then $t_{s-1}(a, i)$ is not defined as this value is defined in the unique treatment of the task (a, i). Thus $t' = t_{s-1} \cup \{(a, i) \mapsto 0\}$ is well defined. Moreover, if there exists a t'-valid oracle $v \supsetneq w_{s-1}$, then step s is clearly possible. Otherwise, by the (sufficiently large) choice of $t_s(a, i)$, the oracle w_{s-1} is even t_s-valid and by Claim 3.5.5, there exists $b \in \{0, 1\}$ such that the oracle $w_s = w_{s-1}b$ is t_s-valid. Hence if some task (a, i) for $a \in \{0, 1\}$ is treated in step s, then we obtain a contradiction.

From now on we assume that step s treats some task $(2,i)$ for $i>0$. Thus $t_s = t_{s-1}$ and we need to show that there exist some t_s-valid $w_s \supsetneq w_{s-1}$ and some $n \in \mathbb{N}$ such that w_s is defined for all words of length n^i+i and

$$0^n \in A^{w_s} \Leftrightarrow P_i^{w_s}(0^n) \text{ rejects.}$$

Choose n large enough such that $2^n > 2\cdot(n^i+i)$. Let $u_0 \sqsupseteq w_{s-1}$ be the minimal t_s-valid oracle that is defined for all words of length $\leq n$. Such an oracle exists by Claim 3.5.5. Moreover, let $u \sqsupseteq u_0$ be the minimal t_s-valid oracle that is defined for all words of length $\leq n^i+i$. Such an oracle exists by Claim 3.5.5 and it holds by Claim 3.5.5.3 that $u \cap 0\Sigma^n = \emptyset$. If $P_i^u(0^n)$ accepts, then it definitely accepts by the choice of u and since $0^n \notin A^v$ for all $v \sqsupseteq u$ (note that u is defined for all words of length $n+1$), the oracle u is t'-valid for $t' = t_{s-1} \cup \{(2,i) \mapsto 0\}$ and hence step s of the construction is possible, a contradiction to our assumption.

From now on we assume that $P_i^u(0^n)$ rejects. Let U be the set of those oracle queries of $P_i^u(0^n)$ whose length is $\geq n+1$. We define $Q_0(U) = U$ and for $m \in \mathbb{N}$

$$Q_{m+1}(U) := \bigcup_{\substack{j\in\mathbb{N}^+,x,y\in\mathbb{N},\\ \{10c(j,x,y),11c(j,x,y)\}\cap Q_m(U)\neq\emptyset}} \{q \in \Sigma^{\geq n+1} \mid q \text{ is queried by } F_j^u(x)\}.$$

Moreover, define $Q(U) = \bigcup_{m\in\mathbb{N}} Q_m(U)$.

Claim 3.5.6 $\ell(Q(U)) \leq 2\cdot(n^i+i)$.

Proof By definition of $Q_0(U)$ it holds $\ell(Q_0(U)) \leq n^i+i$. We show that for all $m \in \mathbb{N}$ it holds $\ell(Q_{m+1}(U)) \leq \ell(Q_m(U))/2$. Then for all $m \in \mathbb{N}$ it holds $\ell(Q_m(U)) \leq \ell(Q_0(U))/2^m$ and thus

$$\ell\Big(\bigcup_{k=0}^{m} Q_k(U)\Big) \leq \ell(Q_0(U)) \cdot \sum_{k=0}^{m} 1/2^k \leq (n^i+i)\cdot\frac{1-1/2^{m+1}}{1/2} < 2\cdot(n^i+i),$$

which shows $\ell(Q(U)) \leq 2\cdot(n^i+i)$.

It remains to show that for all $m \in \mathbb{N}$ it holds $\ell(Q_{m+1}(U)) \leq \ell(Q_m(U))/2$. Let $\alpha \in Q_m(U)$. If α is not of the form $10c(j,x,y)$ or $11c(j,x,y)$, then it generates no elements in $Q_{m+1}(U)$. Assume $\alpha = 1bc(j,x,y)$ for $b \in \{0,1\}$, $j \in \mathbb{N}^+$, and $x,y \in \mathbb{N}$. Then all queries of $F_j^u(x)$ are added into $Q_{m+1}(U)$. The computation time of $F_j^u(x)$ and thus also the sum of the lengths of all queries asked by that computation is bounded by $|x|^j + j \leq |c(j,x,y)|/2$ (cf. the definition of $c(\cdot,\cdot,\cdot)$ and the definition of the pairing function). Hence

$$\begin{aligned}
\ell(Q_{m+1}(U)) &= \ell\Bigg(\bigcup_{\substack{j\in\mathbb{N}^+,x,y\in\mathbb{N},\\ \{10c(j,x,y),11c(j,x,y)\}\cap Q_m(U)\neq\emptyset}} \{q \in \Sigma^{\geq n+1} \mid q \text{ is queried by } F_j^u(x)\}\Bigg)\\
&\leq \sum_{\substack{j\in\mathbb{N}^+,x,y\in\mathbb{N},\\ \{10c(j,x,y),11c(j,x,y)\}\cap Q_m(U)\neq\emptyset}} \underbrace{\ell\big(\{q \in \Sigma^{\geq n+1} \mid q \text{ is queried by } F_j^u(x)\}\big)}_{\leq |c(j,x,y)|/2}\\
&\leq 1/2 \cdot \sum_{\substack{j\in\mathbb{N}^+,x,y\in\mathbb{N},\\ \{10c(j,x,y),11c(j,x,y)\}\cap Q_m(U)\neq\emptyset}} |c(j,x,y)| \leq \ell(Q_m(U))/2,
\end{aligned}$$

which finishes the proof. □

By the choice of n, it holds $|Q(U)| \leq \ell(Q(U)) \leq 2\cdot(n^i+i) < 2^n$. Hence there exists some $\alpha \in 0\Sigma^n$ that is not in $Q(U)$. Let u' be the minimal t_s-valid oracle $\sqsupseteq u_0$ that is defined for all words $\leq 01^n$ and that satisfies $u' \cap 0\Sigma^n = \{\alpha\}$. Such an oracle exists by Claim 3.5.5.3.

Claim 3.5.7 *There exists a t_s-valid oracle $v \sqsupsetneq u'$ that is defined for all words of length $n^i + i$ and satisfies $v(q) = u(q)$ for all $q \in Q(U)$.*

Proof As $\alpha \notin Q(U)$ it holds $u'(q) = u(q)$ for all $q \in Q(U)$ that u' is defined for. Moreover, observe that as $w \sqsupseteq u' \sqsupsetneq u_0$, $u \sqsupseteq u_0$, and u_0 is defined for all words of length $\leq n$, it holds

$$\forall_{q \in \Sigma^{\leq n}}\ w(q) = u(q). \tag{3.26}$$

It suffices to show the following:

> For each t_s-valid partial oracle $w \sqsupseteq u'$ with $w(q) = u(q)$ for all $q \in Q(U)$ that w is defined for, there exists $b \in \{0,1\}$ such that wb is t_s-valid and $wb(q) = u(q)$ for all $q \in Q(U)$ that wb is defined for. (3.27)

Let some $w \in \Sigma^*$ with the properties of (3.27) be given. Moreover, let $z = |w|$, i.e., z is the least word that w is not defined for. We proceed in analogy to the cases of Claim 3.5.5.

1. If Claim 3.5.5.1 (resp., Claim 3.5.5.2) can be applied to w, then we choose $b = 1$ and obtain that wb is t_s-valid. In the present case there exist $a = 0$ (resp., $a = 1$), $j \in \mathbb{N}^+$, and $x, y \in \mathbb{N}$ such that $z = 1ac(j, x, F_j^w(x))$ and $0 < t_s(a, j) \leq z$.

   It remains to prove $z \in Q(U) \Rightarrow z \in u$. If $z \in Q(U)$, then all queries of length $\geq n+1$ of the computation $F_j^u(x)$ are in $Q(U)$. For each such query q it holds $|q| \leq |x|^j + j < |c(j, x, F_j^u(x))| < z = |w|$ and thus by assumption, $w(q) = u(q)$. By that and (3.26), $F_j^u(x) = F_j^w(x)$. As u is t_s-valid and $0 < t_s(a, j) \leq z = 1ac(j, x, F_j^u(x))$, we obtain by V3/V5 (depending on the value of a) that $z = 1ac(j, x, F_j^u(x)) \in u$.

2. If Claim 3.5.5.3 can be applied to w, then we choose $b = u(z) = 0$ (recall the definition of u for $u(z) = 0$) and obtain that wb is t_s-valid. By the definition of b, it holds $wb(q) = u(q)$ for all $q \in Q(u)$ that wb is defined for.

3. If Claim 3.5.5.4 can be applied to w, then we choose $b = 0$ and obtain that wb is t_s-valid. Note that in this case $z \in 1\Sigma^*$.

   It remains to show that $z \in Q(U) \Rightarrow z \notin u$. For a contradiction assume $z \in Q(U) \cap u$. Let u'' be the prefix of u that is defined for exactly the words $< z$. As $w_{s-1} \sqsubseteq u'' \sqsubseteq u$ and both w_{s-1} and u are t_s-valid, u'' is t_s-valid as well (cf. Claim 3.5.4).

   - Assume Claim 3.5.5.1 or Claim 3.5.5.2 is applicable to u'', thus $z = 1ac(j, x, F_j^{u''}(x))$ for $a \in \{0,1\}$, $j \in \mathbb{N}^+$, and $x \in \mathbb{N}$ with $0 < t_s(a, j) \leq z$. By Claim 3.5.2, $F_j^u(x) = F_j^{u''}(x)$, which implies $F_j^u(x) \neq F_j^w(x)$ (otherwise, $F_j^w(x) = F_j^{u''}(x)$ and either Claim 3.5.5.1 or Claim 3.5.5.2 can be applied to w, contradicting the assumption that Claim 3.5.5.4 can be applied to w). Hence there is some query $q \in u \triangle w$ that is asked by both computations $F_j^u(x)$ and $F_j^w(x)$ (otherwise, the two computations would output the same value). Note that $q \in Q(U)$ or $|q| \leq n$, since (i) $z = 1ac(j, x, F_j^{u''}(x)) \in Q(U)$ by assumption and (ii) q is queried by the computation $F_j^u(x)$. As $|q| \leq |x|^j + j < |c(j, x, F_j^u(x))| < c(j, x, F_j^u(x)) = z$, the oracle w is defined for q and thus by assumption and (3.26), $w(q) = u(q)$, a contradiction.
   - Now assume that neither Claim 3.5.5.1 nor Claim 3.5.5.2 can be applied to u''. Then, as $z \in 1\Sigma^*$, Claim 3.5.5.4 is applicable to u'' and $u''0$ is t_s-valid. By Claim 3.5.5, $u''0$ can be extended to a t_s-valid oracle $\widetilde{u}$ defined for exactly the words of length $\leq n^i + i$. As u and $\widetilde{u}$ agree on all words $< z$ and $\widetilde{u}(z) = 0 < 1 = u(z)$, it holds $\widetilde{u} < u$, a contradiction to the choice of u.

In both cases we obtain a contradiction. Hence $u(q) = w0(q)$ for all $q \in Q(U)$ that $w0$ is defined for.

In all cases (3.27) holds. This completes the proof of Claim 3.5.7. □

Recall that in the case we are currently studying the computation $P_i^u(0^n)$ rejects. Let v be the oracle whose existence is postulated by Claim 3.5.7. Since all queries of $P_i^u(0^n)$ are in $U \subseteq Q(U)$ and by Claim 3.5.7, u and v agree on all these queries, the computation $P_i^v(0^n)$ rejects as well. Moreover, this computation is definite as v is defined for all words of length $n^i + i$. However, as $\alpha \in v$, we obtain $0^n \in A^{v'}$ for all $v' \sqsupseteq v$, which is a contradiction to the assumption that the construction fails in step s treating the task $(2, i)$.

We now have seen that the construction described above is possible. It remains to prove:

- $\mathrm{NP}^O \neq \mathrm{P}^O$,
- $\mathcal{CAN}^O$ has P^O-optimal proof systems, and
- $\overline{\mathcal{CAN}^O}$ has P^O-optimal proof systems.

This is shown in the next three claims.

Claim 3.5.8 $\mathrm{NP}^O \neq \mathrm{P}^O$.

Proof Assume $\mathrm{NP}^O = \mathrm{P}^O$. Then there exists $i > 0$ such that $L(P_i^O) = A^O$. Let s be the step with $T(s) = (2, i)$. By construction, there exists some $n \in \mathbb{N}$ such that w_s is defined for all words of length $n^i + i$ and it holds $\big(0^n \in A^{w_s} \Leftrightarrow P_i^{w_s}(0^n) \text{ rejects}\big)$. As w_s is defined for all words of length $n^i + i$, it holds $A^{w_s}(0^n) = A^v(0^n)$ for all $v \sqsupseteq w_s$ and $P_i^{w_s}(0^n)$ definitely rejects. Hence $0^n \in A^O$ if and only if $P_i^O(0^n)$ rejects, which contradicts $L(P_i^O) = A^O$. Thus $\mathrm{NP}^O \neq \mathrm{P}^O$. □

Claim 3.5.9 *$\mathcal{CAN}^O$ has P^O-optimal proof systems.*

Proof Let g be a proof system for $\mathcal{CAN}^O$ and $a \in \mathcal{CAN}^O$. Define

$$f(z) = \begin{cases} y & \text{if } z = 010c(i,x,y) \text{ and } 10c(i,x,y) \in O \text{ for } i \in \mathbb{N}^+ \text{ and } x, y \in \mathbb{N} \\ g(y) & \text{if } z = 1y \\ a & \text{otherwise} \end{cases}$$

Then $f \in \mathrm{FP}^O$ and $f(\mathbb{N}) \supseteq \mathcal{CAN}^O$ as g is a proof system for $\mathcal{CAN}^O$. We show $f(\mathbb{N}) \subseteq \mathcal{CAN}^O$. As g is a proof system for $\mathcal{CAN}^O$ and $a \in \mathcal{CAN}^O$, it suffices to show $f(z) \in \mathcal{CAN}^O$ for $z = 010c(i,x,y)$ with $10c(i,x,y) \in O$, $i \in \mathbb{N}^+$, and $x, y \in \mathbb{N}$. Let s be large enough such that w_s is defined for $10c(i,x,y)$. Then by V1 and Claim 3.5.2, $y \in \mathcal{CAN}^v$ for all $v \sqsupseteq w_s$. It follows $f(z) = y \in \mathcal{CAN}^O$ and thus f is a proof system for $\mathcal{CAN}^O$.

In order to show that f is P^O-optimal, let h be an arbitrary proof system for $\mathcal{CAN}^O$. Then there exists $i \in \mathbb{N}^+$ such that F_i^O computes h. Let s be the step with $T(0, i) = s$. It holds $t_s(0, i) > 0$ (otherwise, by V2 there exists x such that $F_i^w(x)$ is definite and $F_i^w(x) \notin \mathcal{CAN}^v$ for all $v \sqsupseteq w$, which would imply that F_i^O is not a proof system for $\mathcal{CAN}^O$). Define

$$\pi(x) = \begin{cases} 010c(i, x, F_i^O(x)) & \text{if } 10c(i, x, F_i^O(x)) \geq t_s(0, i) \\ z & \text{if } 10c(i, x, F_i^O(x)) < t_s(0, i) \text{ and } z \text{ is minimal with } f(z) = F_i^O(x) \end{cases}$$

π is total as f and F_i^O are proof systems for $\mathcal{CAN}^O$ and thus for each x there exists z with $f(z) = F_i^O(x)$. Moreover, since $t_s(0,i)$ is a constant, $\pi \in \mathrm{FP}^O$. It remains to show $F_i^O(x) = f(\pi(x))$ for all x. For all x with $10c(i,x,F_i^O(x)) < t_s(0,i)$ this clearly holds. Assume $10c(i,x,F_i^O(x)) \geq t_s(0,i)$. Choose $s' \geq s$ large enough such that $w_{s'}$ is defined for $10c(i,x,F_i^O(x))$ and $F_i^{w_{s'}}(x)$ is definite. Hence $F_i^{w_{s'}}(x) = F_i^O(x)$. Then by V3, $10c(i,x,F_i^O(x)) = 10c(i,x,F_i^{w_{s'}}(x)) \in w_{s'} \subseteq O$ and thus $f(\pi(x)) = f(010c(i,x,F_i^O(x))) = F_i^O(x)$. □

The following claim can be proven in an analogous way.

Claim 3.5.10 *$\overline{\mathcal{CAN}^O}$ has P^O-optimal proof systems.*

This completes the proof of Theorem 3.5.1. □

Corollary 3.5.11 *There exists an oracle O relative to which the following statements hold:*

1. $\mathrm{P}^O \neq \mathrm{NP}^O$
2. *Each non-empty set in NP^O has P^O-optimal proof systems.*
3. *Each non-empty set in coNP^O has P^O-optimal proof systems.*

Proof The statement follows from Theorem 3.5.1 and Corollary 2.3.6. □

3.6 Summary and Discussion

Taking also into account the aforementioned unpublished results by Fabian Egidy, Anton Ehrmanntraut, and Christian Glaßer, Figure 3.1 illustrates the current situation regarding the known implications and oracle separation between the conjectures introduced at the beginning of the present chapter. Thus in contrast to Figure 1.2, Figure 3.1 contains not only the published, but also the unpublished results we are aware of.

So for most pairs (A,B) of conjectures from Figure 3.1 we either know a relativizable proof for the implication $\mathsf{A} \Rightarrow \mathsf{B}$ or know that there is an oracle relative to which $\mathsf{A} \wedge \neg\mathsf{B}$. This would hold for *all* pairs (A,B) of conjectures if three more oracles with the below properties were constructed:

1. $\mathsf{UP} \wedge \neg\mathsf{CON}^{\mathsf{N}}$
2. $\mathsf{SAT} \wedge \neg\mathsf{TFNP}$
3. $\mathsf{TFNP} \wedge \neg\mathsf{DisjCoNP}$.

Therefore, we are interested in the following question: Do oracles with these properties exist?

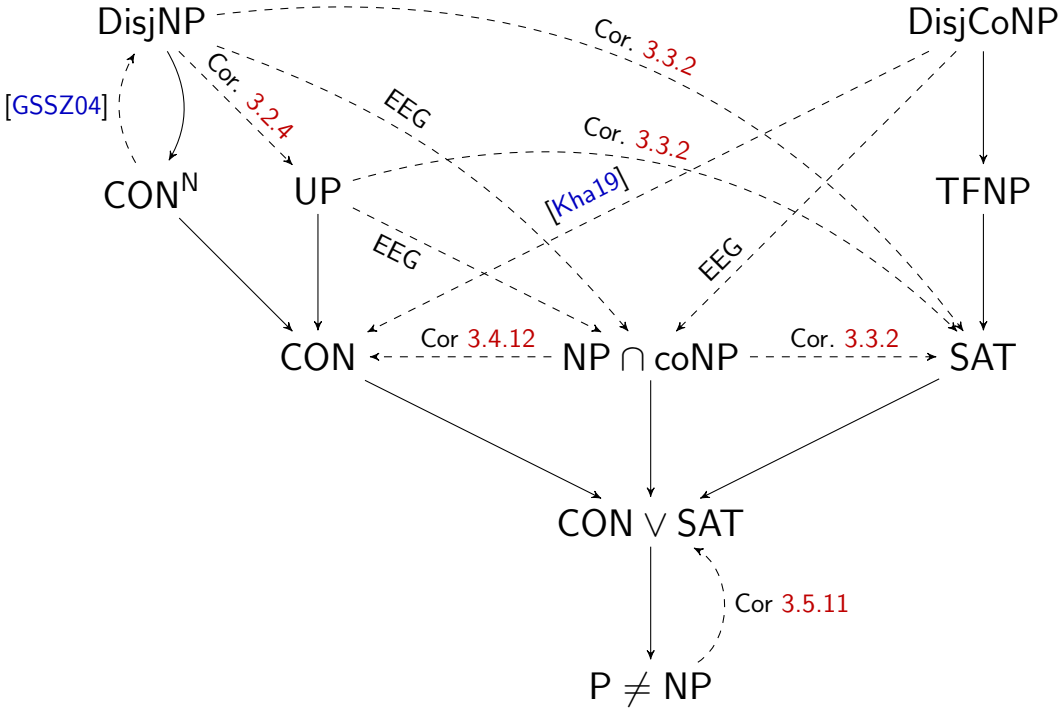


Figure 3.1: Solid arrows mean implications. A dashed arrow from one conjecture A to another conjecture B means that there is an oracle X against the implication $\mathsf{A} \Rightarrow \mathsf{B}$, which means $\mathsf{A} \wedge \neg \mathsf{B}$ relative to X. EEG stands for unpublished results by Fabian Egidy, Anton Ehrmanntraut, and Christian Glaßer.

Chapter 4

Balance Problems for Integer Circuits

In this chapter we introduce balance problems for integer circuits. After defining the notion of balanced sets and the notion of integer circuits in Section 4.1, we prove this chapter's main result, the undecidability of the balance problem which allows both set difference and multiplication, in Section 4.2. Having obtained this undecidability result when the set of allowed operations $\mathcal{O}$ equals $\{-, \cdot\}$, it suggests itself to ask whether even a proper subset of $\mathcal{O}$ suffices to gain undecidability. Therefore, Section 4.3 considers the two problems allowing only one operation, either multiplication or set difference. For the sake of completeness, in the same section we also consider the simple balance problem which does not allow any operations and thus obtain a precise classification of the computational complexity of the three balance problems admitting some $\mathcal{O}' \subsetneq \mathcal{O}$ as the set of allowed operations. Each of these problems turns out to be $\leq_{\mathrm{m}}^{\log}$-complete for one of the classes L, NL, and NP. Section 4.4 concludes this chapter with a brief summary and discussion of the results.

4.1 Basic Definitions and Results

This section defines the concepts the present chapter is based on: the notion of balanced sets and the notion of integer circuits. Moreover, some simple results are obtained.

4.1.1 Balanced Sets

In this subsection we define the notion of balance, show that the test of whether some input set is balanced can be performed by a deterministic logarithmic-space Turing machine, and make some technical observations that follow from the definition of balanced sets.

A finite and non-empty set $S \subseteq \mathbb{N}$ is *balanced* (resp., *unbalanced*) if $|S| = |\{0, 1, \ldots, \max(S)\} - S|$ (resp., $|S| \neq |\{0, 1, \ldots, \max(S)\} - S|$). Intuitively speaking, $\max(S)$ defines the universe $\{0, 1, \ldots, \max(S)\}$ and then S is balanced if it contains the same number of elements as its complement. Note that the notion of balanced and unbalanced sets only makes sense if there is some maximum element defining the universe. Hence the empty set is neither balanced nor unbalanced. The following lemma follows directly from the definition.

Lemma 4.1.1 *Let $S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ be balanced. Then $S \neq \emptyset$, $\max(S)$ is odd, and $|S| = {}^{(\max(S)+1)}/_{2}$.*

Moreover, we say that $S \neq \emptyset$ is *subbalanced* if $|S| < (\max(S)+1)/2$. This means that S contains less than half of the elements in the universe $\{0, 1, \ldots, \max(S)\}$.

Let us observe that it is possible for a deterministic logarithmic-space Turing machine to test whether some input set is balanced or not. Define $\mathrm{BAL} = \{S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \mid S \text{ is balanced}\}$. We want to prove $\mathrm{BAL} \in \mathrm{L}$ and show an even stronger result, which we will need in Section 4.3. For that purpose we introduce a more general problem. For a finite and non-empty set M let $\mathrm{BAL}_M = \{S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \mid M \cdot S \text{ is balanced}\}$.

Proposition 4.1.2 *For all $M \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$, $\mathrm{BAL}_M \in \mathrm{L}$. In particular, $\mathrm{BAL} \in \mathrm{L}$.*

Proof The second statement follows from the first as $\mathrm{BAL} = \mathrm{BAL}_{\{1\}}$. Assume $M \neq \emptyset \wedge \mu := \max(M) > 0$ (otherwise, $\mathrm{BAL}_M = \emptyset \in \mathrm{L}$). The following algorithm decides BAL_M when given a finite set $S \subseteq \mathbb{N}$ as input, where we assume $S \neq \emptyset \wedge \sigma := \max(S) > 0$ (otherwise, $S \notin \mathrm{BAL}_M$).

1. If $\sigma < 5$, then accept if $S \in \mathrm{BAL}_M$ and reject otherwise.
2. Let $n = \ell(S)$ and $c = 0$. Reject if $2 \cdot (\log n + 1) < |\sigma|$.
3. For $\alpha = 0, 1, \ldots, \mu \cdot \sigma$: if there exists $(m, s) \in M \times S$ with $ms = \alpha$, then increment c.
4. If $2c = \mu \cdot \sigma + 1$, then accept. Otherwise, reject.

If step 3 is executed, then $|\sigma| \leq 2 \log n + 2$ and thus $|\mu \cdot \sigma|$ is logarithmically bounded in $|S|$. Hence the algorithm can be implemented by a deterministic logarithmic-space Turing machine.

Let us now show that the algorithm accepts if and only if $S \in \mathrm{BAL}_M$. This clearly holds if the algorithm terminates in step 1. If the algorithm rejects in step 2, then $\sigma \geq 5$ (cf. step 1) and $2(\log n + 1) < |\sigma|$, where the latter implies $2^{\log n + 1} < \sqrt{2^{|\sigma|}}$. It follows

$$|M \cdot S| \leq \mu \cdot |S - \{0\}| + 1 \leq \mu n + 1 \leq \mu(n+1) \leq \mu 2^{\log n + 1} < \mu \cdot \sqrt{2^{|\sigma|}} \leq \mu \cdot \sqrt{\sigma + 1} \leq \mu \cdot \sigma/2,$$

which shows that $M \cdot S$ is subbalanced. If step 4 is executed, then $c = |M \cdot S|$ during the execution of this step and thus the algorithm accepts or rejects correctly. □

Let us continue with a technical lemma that directly builds on the definition of balanced sets and will be applied several times.

Lemma 4.1.3 *The following statements hold.*

1. *For $K \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ with $\kappa := \max(K) \geq 3$ it holds $|K \cdot K \cdot K| < \kappa^3/2$.*
2. *For all $M, K \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ with $\max(K) \geq 2$, the set $M \cdot K \cdot K \cdot K$ is not balanced.*

Proof 1. Due to $K \cdot K \cdot K \subseteq \{0\} \cup \{i \cdot j \cdot k \mid 1 \leq i \leq j \leq k \leq \kappa\}$ it holds

$$\begin{aligned}
|K \cdot K \cdot K| &\leq 1 + \big|\{(i,j,k) \mid 1 \leq i \leq j \leq k \leq \kappa\}\big| \\
&\leq 1 + \big|\{(i,j,k) \mid 1 \leq i = j = k \leq \kappa\}\big| + \big|\{(i,j,k) \mid 1 \leq i = j < k \leq \kappa\}\big| \\
&\quad + \big|\{(i,j,k) \mid 1 \leq i < j = k \leq \kappa\}\big| + \big|\{(i,j,k) \mid 1 \leq i < j < k \leq \kappa\}\big| \\
&\leq 1 + \kappa + \binom{\kappa}{2} + \binom{\kappa}{2} + \binom{\kappa}{3} = \frac{\kappa^3 + 3\kappa^2 + 2\kappa + 6}{6} < \frac{\kappa^3}{2}.
\end{aligned}$$

2. If $M = \emptyset$ and $M = \{0\}$, it holds that $M \cdot K \cdot K \cdot K = M$ is not balanced. Assume $\max(M) \geq 1$. If $\max(K) = 2$, then $M \cdot K \cdot K \cdot K$ has an even maximum and thus is unbalanced by Lemma 4.1.1. If $\max(K) \geq 3$, then the first statement of the present lemma yields $|M \cdot K \cdot K \cdot K| \leq |(M - \{0\}) \cdot \prod_{i=1}^{3} (K \cup \{0\})| < \max(M) \cdot \max(K)^3/2$ and thus $M \cdot K \cdot K \cdot K$ is unbalanced. □

4.1.2 Integer Circuits and Balance Problems

As in the present chapter we consider the computational complexity of balance problems with respect to the logarithmic-space many-one reducibility $\leq_{\mathrm{m}}^{\log}$, it is convenient to ensure that the test of whether some input tuple is a valid circuit can be performed by a deterministic logarithmic-space Turing machine. Hence when defining the concept of integer circuits we will keep an eye on this issue, which has already been taken care of in articles such as [GHR+10, GRTW10, BBD+17, BBD+20]. Therefore, we only need to follow these papers making minor adaptions, namely introducing gates for the set difference and the symmetric difference instead of the set complement and allowing input gates to compute arbitrary finite subsets of $\mathbb{N}$ and not only singleton sets.

Definition of Circuits Barth et al. [BBD+20] differentiate between completely and partially assigned circuits. As in this thesis we restrict ourselves to partially assigned circuits, we define circuits in general as partially assigned circuits.[1]

A *circuit* C is a triple (V, E, g_C) where (V, E) is a directed acyclic multigraph, g_C is a designated vertex in V, and the vertex set $V \subseteq \mathbb{N}$ is topologically ordered, i.e., for all vertices $u, v \in V$ with $u < v$ there is no edge from v to u. Recall that according to Section 2.2 the multigraph (V, E) is finite, non-empty, and not necessarily connected.

Let us explain why we require that circuits have topologically ordered vertex sets. The test of whether some directed multigraph is acyclic is not known to be logarithmic-space computable. However, it is straightforward to construct a deterministic logarithmic-space Turing machine which tests whether some directed multigraph has a topologically ordered vertex set. Moreover, each directed multigraph with topologically ordered vertex set is acyclic. Thus a deterministic logarithmic-space Turing machine is able to check whether some input tuple is a circuit. Hence when presenting algorithms for circuits we may always assume that the input tuple is a valid circuit.

Without loss of generality, it may be assumed that $V = \{1, \ldots, r\}$ for some $r \in \mathbb{N}$ (otherwise, renumber the circuit, which is possible in logarithmic space).

In the context of circuits, the terms "node" and "gate" are used synonymously. A gate with indegree 0 is called an *input gate* (resp., *input node*), all other nodes are *inner gates* (resp., *inner nodes*), and the designated gate g_C is also called an *output gate* (resp., *output node*).

Let $\mathcal{O} \subseteq \{\triangle, -, \cup, \cap, +, \cdot\}$. An *$\mathcal{O}$-circuit* (or *circuit* for short if $\mathcal{O}$ is apparent from the context) is a quintuple $C = (V, E, g_C, \alpha, \beta)$ satisfying the following five properties.

- (V, E, g_C) is a circuit which satisfies the property that each node has indegree 0 or 2.
- The *labeling function* α is a total function $V \to \mathcal{O} \cup \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \cup \{\square\}$.
- Each node with indegree 0 has a label from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ or from $\{\square\}$.
- Nodes with indegree 2 have labels from $\mathcal{O}$.
- The *order function* β is a total function $E \to \{\mathrm{l}, \mathrm{r}\}$ satisfying the property that for each node u with indegree 2 (i.e., each inner node) it holds $\beta(e) \neq \beta(e')$ for the two incoming edges e and e' of u.

Thus for each gate with indegree 2 the order function tells us which is the left and which is the right direct predecessor (note that left and right direct predecessor may be equal). Clearly the

[1] In parts this paragraph and also the next two paragraphs, entitled as "The Set Computed by a Circuit" and "Basic Constructions", orientate themselves by corresponding paragraphs from our paper [BBD+20].

order function is only relevant for nodes u with $\alpha(u) = -$. In particular, if $- \notin \mathcal{O}$, we may go without an order function.

Let us say a few words on the encoding of $\mathcal{O}$-circuits. The set of edges is encoded as a list and we accept those lists as part of an encoding of a given circuit in which for each inner node $u \in V$ the incoming edge e of u with $\beta(e) = \mathrm{l}$ occurs earlier than the incoming edge e' of u with $\beta(e') = \mathrm{r}$. Thus the order function β is implicitly contained in the list of edges and the encoding of a circuit does not contain an explicit representation of the order function β. Moreover, the objects in $\{\triangle, -, \cup, \cap, +, \cdot, \square\}$ are encoded by words of length 3. Thus recalling the paragraph about encodings in Section 2.3.2, we observe that now an encoding of $\mathcal{O}$-circuits is defined and all possible encodings of a certain $\mathcal{O}$-circuit have the same length.

$\mathcal{O}$-circuits are also called *integer circuits* or *partially assigned $\mathcal{O}$-circuits*. Input gates labeled with an element of $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ are *assigned input gates* (resp., *assigned input nodes*), whereas input gates labeled with $\square$ are called *unassigned input gates* (resp., *unassigned input nodes*). For $\oplus \in \mathcal{O}$ we call an inner gate v with $\alpha(v) = \oplus$ a *$\oplus$-gate* (resp., *$\oplus$-node*).

If g is a gate with $\alpha(g) = \oplus \in \mathcal{O}$, the (not necessarily distinct) nodes g_1 and g_2 are the source nodes of the two incoming edges e_1 and e_2 of g, and the two equations $\beta(e_1) = \mathrm{l}$ and $\beta(e_2) = \mathrm{r}$ hold, then we write $g = g_1 \oplus g_2$. Note that it is important to consider the order of the operands if and only if $\oplus = -$.

Observe that there exists a straightforward deterministic logarithmic-space algorithm that decides whether an input tuple is a valid $\mathcal{O}$-circuit.

The Set Computed by a Circuit For an $\mathcal{O}$-circuit C with unassigned input gates $g_1 < \cdots < g_n$ and $X_1, \ldots, X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$, let $C(X_1, \ldots, X_n)$ be the integer circuit that arises from C by modifying the labeling function α such that $g_i \mapsto X_i$ for every $1 \leq i \leq n$.

For an $\mathcal{O}$-circuit $C = (V, E, g_C, \alpha, \beta)$ without unassigned input gates we inductively define the set $I(g; C)$ computed by a gate $g \in V$:

$$I(g; C) = \begin{cases} \alpha(g) & \text{if } g \text{ has indegree } 0, \\ I(g', C) \oplus I(g'', C) & \text{if } g = g' \oplus g'' \text{ for some } \oplus \in \mathcal{O}. \end{cases}$$

The set computed by the circuit is denoted by $I(C)$ and defined to be the set computed by the output gate $I(g_C; C)$.

We use the following abbreviations assuming no confusions will arise: for an integer circuit C without unassigned input gates and a gate g of C we write g or $I(g)$ for $I(g; C)$ if C is apparent from the context; moreover, we write C for $I(C)$.

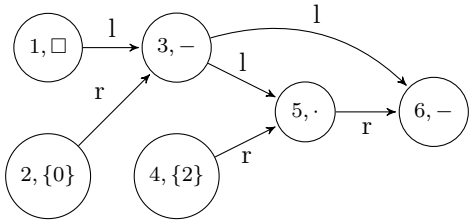
Basic Constructions for $\mathcal{O}$-Circuits We often construct or modify circuits successively. For that purpose we introduce some modes of speaking. Let $C = (V, E, g_C, \alpha, \beta)$ and $C' = (V', E', g_{C'}, \alpha', \beta')$ be $\mathcal{O}$ circuits.

By saying that we add (or insert) C' into C we mean that we unify the circuits and let g_C be the new circuit's output gate, i.e., we first renumber the nodes of C' such that V' is disjoint to V and then create the new circuit $(V \cup V', E \cup E', g_C, \alpha'', \beta'')$ with $\alpha'' \colon V \cup V' \to \mathcal{O} \cup \{\square\} \cup \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ given by $x \mapsto \alpha(x)$ for $x \in V$ and $x \mapsto \alpha'(x)$ for $x \in V'$ and $\beta'' \colon E \cup E' \to \{\mathrm{l}, \mathrm{r}\}$ defined analogously. We denote the circuit obtained this way by $C = (V, E, g_C, \alpha, \beta)$ as well. Note that this operation can be performed by a deterministic logarithmic-space Turing machine.

Moreover, we use the self-explaining phrase "add (resp., insert or introduce) nodes (and edges) into C such that there is a gate $g = F(g_1, \ldots, g_n)$" where $g_1, \ldots, g_n$ are gates in C and F is an expression built up from the gates given as arguments, the operations from $\mathcal{O}$, and parentheses. Instead of this, we may also use the less precise phrase "add (resp., insert or

introduce) a node $g = F(g_1, \ldots, g_n)$ (into C)" even though there might be more than one node to add in order to obtain such a node.

Example As an example, consider the following circuit



where each node is given by its number and its label and node 6 is the unique output gate. Node 1 is an unassigned input node, which is marked by the symbol □. Observe that the output gate computes the set $\{1\}$ if and only if $I(3; C)$ is a set of the form $\{2^0, 2^1, 2^2, \ldots, 2^r\}$ for some $r \in \mathbb{N}$ (a more general statement will be proven later in detail).

The Main Problems Now we define the problems this chapter focuses on.

Definition 4.1.4 *Let $\mathcal{O} \subseteq \{\triangle, -, \cup, \cap, +, \cdot\}$ and define*

$$\mathrm{BC}(\mathcal{O}) = \{C \mid C \text{ is an } \mathcal{O}\text{-circuit with } n \in \mathbb{N} \text{ unassigned inputs and there exist } X_1, \ldots, X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \text{ such that } I(C(X_1, \ldots, X_n)) \text{ is balanced}\}.$$

Moreover, for an $\mathcal{O}$-circuit C with $n \in \mathbb{N}$ unassigned input gates we call $X_1, \ldots, X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ an assignment *(for C). Furthermore, an assignment for a circuit is called* balancing *if the circuit is balanced under this assignment.*

Note that formally $\mathrm{BC}(\mathcal{O})$ must be defined as a set of words or natural numbers. Nevertheless, as has been announced before, we ignore this technical detail here. Moreover, note that we will use shortcuts like $\mathrm{BC}(-, \cdot)$ instead of $\mathrm{BC}(\{-, \cdot\})$.

The following proposition holds as by definition, $\mathrm{BC}(\mathcal{O})$ is a projection of a decidable set.

Proposition 4.1.5 *For $\mathcal{O} \subseteq \{\triangle, -, \cup, \cap, +, \cdot\}$ it holds $\mathrm{BC}(\mathcal{O}) \in \mathrm{RE}$.*

In this thesis we focus for the most part on the problems $\mathrm{BC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{-, \cdot\}$. In order to prove $\mathrm{BC}(\cdot)$ to be $\leq_{\mathrm{m}}^{\log}$-hard for NL, we will make use of the following problem investigated by McKenzie and Wagner [MW07], which they show to be $\leq_{\mathrm{m}}^{\log}$-complete for NL.[2]

$$\mathrm{MC}(\cap) = \{(C, b) \mid C \text{ is an } \{\cap\}\text{-circuit whose inputs are all assigned and have labels from } \{X \subseteq \mathbb{N} \mid |X| = 1\},\ b \in I(C)\}.$$

The following lemma follows from Definition 4.1.4.

Lemma 4.1.6 *For $\mathcal{O} \subseteq \mathcal{O}'$ it holds $\mathrm{BC}(\mathcal{O}) \leq_{\mathrm{m}}^{\log} \mathrm{BC}(\mathcal{O}')$.*

Therefore, each lower (resp., upper) bound for a problem $\mathrm{BC}(\mathcal{O})$ implies the same lower (resp., upper) bound for all problems $\mathrm{BC}(\mathcal{O}')$ with $\mathcal{O}' \supseteq \mathcal{O}$ (resp., $\mathcal{O}' \subseteq \mathcal{O}$).

[2] We define the problem within the notions and notations we have introduced above and which are not precisely the same as those of McKenzie and Wagner. Nevertheless, the NL-completeness of the problem $\mathrm{MC}(\cap)$ (defined the way we do it) with respect to $\leq_{\mathrm{m}}^{\log}$ follows from their paper.

4.2 Set Difference and Multiplication Lead to Undecidability

In this section we prove this chapter's main result: the undecidability of $\mathrm{BC}(-,\cdot)$. As a trivial corollary we obtain that $\mathrm{BC}(\triangle,\cup,\cdot)$ and $\mathrm{BC}(\triangle,\cap,\cdot)$ are undecidable as well. Indeed, all these problems are even $\leq_{\mathrm{m}}$-complete for RE.

We briefly introduce the notion of multivariate polynomials. Let $x_1, x_2, \ldots$ be fixed symbols, which we call *indeterminates* in the context of multivariate polynomials.

For $m, a_1, \ldots, a_m \in \mathbb{N}^+$, $n \in \mathbb{N}$, and $d_{i,j} \in \mathbb{N}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ we call

$$p = \sum_{i=1}^{m} a_i \cdot \prod_{j=1}^{n} x_j^{d_{i,j}}$$

a *multivariate polynomial*. We also write $p(x_1, \ldots, x_n)$ for the multivariate polynomial p. Each $a_i \cdot \prod_{j=1}^{n} x_j^{d_{i,j}}$ is called a *multivariate monomial*. We identify each multivariate polynomial $\sum_{i=1}^{m} a_i \cdot \prod_{j=1}^{n} x_j^{d_{i,j}}$ with the multivariate polynomial $\sum_{i=1}^{m} a_i \cdot \prod_{j=1}^{n+1} x_j^{d_{i,j}}$ if $d_{i,n+1} = 0$ for all i. Hence for the above polynomial p we may also write $p(x_1, \ldots, x_r)$ where r is an arbitrary natural number $\geq n$. Regarding the encoding the multivariate polynomial p can be considered as the tuple

$$(n, a_1, , d_{1,1}, \ldots, d_{1,n}, a_2, d_{2,1}, \ldots, d_{2,n}, \ldots, a_m, d_{m,1}, \ldots, d_{m,n})$$

where we choose $n = \max(\{j \mid \exists_{i \in \{1,\ldots,m\}}\, d_{i,j} > 0\})$ (so roughly speaking, we ignore redundant variables, which only occur with exponent 0). For the encoding of tuples we refer to the paragraph about encodings in Section 2.3.2.

For each $m \geq n$ a multivariate polynomial $p(x_1, \ldots, x_n)$ induces a function $\mathbb{N}^m \to \mathbb{N}$ defined by $(a_1, \ldots, a_m) \mapsto p(a_1, \ldots, a_m)$ where $p(a_1, \ldots, a_m)$ is the natural number that we obtain when replacing in p each occurrence of x_i with a_i and evaluating the expression generated this way over $\mathbb{N}$. In the context of functions we call the indeterminates variables. We identify a multivariate polynomial with its induced functions and hence we may use the terms "indeterminate" and "variable" interchangeably.

An equation $p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$ for multivariate polynomials $p(x_1, \ldots, x_n)$ and $q(x_1, \ldots, x_n)$ is called a *Diophantine equation* and $(a_1, \ldots, a_n) \in \mathbb{N}^n$ is a solution of the Diophantine equation if $p(a_1, \ldots, a_n) = q(a_1, \ldots, a_n)$.[3] According to the Matiyasevich-Robinson-Davis-Putnam theorem [Mat70, DPR61] the problem of determining whether there is a solution for a given Diophantine equation is RE-complete with respect to $\leq_{\mathrm{m}}$. It can be derived by standard arguments that the following problem is $\leq_{\mathrm{m}}$-complete for RE as well.

$$\begin{aligned}\mathcal{DE} = \{(p,q) \mid\ & p(x_1, \ldots, x_n) \text{ and } q(x_1, \ldots, x_n) \text{ for some } n \in \mathbb{N}^+ \text{ are multivariate polynomials,} \\ & \exists_{a_1,\ldots,a_n \in \mathbb{N}^+}\, p(a_1, \ldots, a_n) = q(a_1, \ldots, a_n)\}.\end{aligned}$$

Reducing this problem to $\mathrm{BC}(-,\cdot)$ will prove the following theorem.

Theorem 4.2.1 $\mathrm{BC}(-,\cdot)$ *is* $\leq_{\mathrm{m}}$*-complete for* RE.

The remainder of this section is almost completely dedicated to the purpose of proving this theorem. The proof consists of several lemmas.

[3] Note that the problem of determining whether a Diophantine equation has a solution is known to be $\leq_{\mathrm{m}}^{\log}$-equivalent to the corresponding problem when Diophantine equations and their solutions are defined over the set of integers instead of the set of natural numbers. The reduction $\leq_{\mathrm{m}}^{\log}$ follows from Lagrange's four-square theorem and the opposite reduction can be shown by replacing each integer variable with the difference of two "natural variables", expanding, and moving negative monomials to the opposite side.

For the sake of brevity, we make use of intersection gates but note that $A \cap B$ is just an abbreviation for $A-(A-B)$. Further abbreviated notations are $A-\bigcup_{i=1}^{n} B_i$ for $(\dots((A-B_1)-B_2)-\dots)-B_n$ and $A-(\bigcup_{i=1}^{n} B_i-\{1\})$ for $(\dots((A-(B_1-\{1\}))-(B_2-\{1\}))-\dots-(B_n-\{1\})$.

In order to prove Theorem 4.2.1, we define a slightly different version of the problem $\mathrm{BC}(-,\cdot)$, which can be $\leq_{\mathrm{m}}$-reduced to the original version.

Definition 4.2.2 *For $\mathcal{O} \subseteq \{\triangle, -, \cup, \cap, +, \cdot\}$ we define the following problem.*

$$\mathrm{BC}'(\mathcal{O}) = \big\{(C,Q) \mid C=(V,E,g_c,\alpha,\beta) \text{ is an } \mathcal{O}\text{-circuit, } \mathrm{ran}(\alpha) \subseteq \mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+) \cup \mathcal{O} \cup \{\square\},\ Q \subseteq V,\\ \exists_{X_1,\dots,X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)}\big(C(X_1,\dots,X_n) \text{ is balanced } \wedge \forall_{K\in Q}\, I(K; C(X_1,\dots,X_n)) = \{1\}\big)\big\}$$

For the sake of simplicity, instances of $\mathrm{BC}'(\mathcal{O})$ are called $\mathcal{O}$-circuits as well.

So the differences from $\mathrm{BC}(\mathcal{O})$ basically are that the assigned inputs of a circuit compute sets that do not contain 0 and that we not only require the existence of some balancing assignment, but the existence of a balancing assignment under which two additional requirements are satisfied, namely (i) all assigned inputs compute sets not containing 0 and (ii) the nodes in the designated set of vertices all compute $\{1\}$.

Lemma 4.2.3 *For $\{-,\cdot\} \subseteq \mathcal{O} \subseteq \{\triangle, -, \cup, \cap, +, \cdot\}$ it holds $\mathrm{BC}'(\mathcal{O}) \leq_{\mathrm{m}} \mathrm{BC}(\mathcal{O})$.*

Proof We described the required reduction function. Let C be a partially assigned $\mathcal{O}$-circuit with $n \in \mathbb{N}$ unassigned input gates and an output node g_C. Let Q be a subset of the nodes of C. Without loss of generality, we assume that g_C is not an input node of the circuit (otherwise, introduce new nodes and edges such that there is a new output $o = g_C \cdot \{1\}$) and that $0 \notin \alpha(g)$ for all assigned input gates g (otherwise, $(C,Q) \notin \mathrm{BC}'(\mathcal{O})$ and we can map (C,Q) to some fixed circuit that is not in $\mathrm{BC}(\mathcal{O})$). Our reduction has to address (i) the nodes in Q and (ii) the fact that for the membership in $\mathrm{BC}'(\mathcal{O})$ we only care about assignments $X_1,\dots,X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$. Starting with the circuit C, we build a new circuit as follows and denote this modified circuit with C':

1. For each unassigned input node g:
   (a) Add a node $g' = g - \{0\}$.
   (b) Except for the edge from g to g', let all outgoing edges of g start in g' instead.
2. Insert nodes and edges such that there is a new output node $g_{C'} = g_C \cdot \prod_{K\in Q}(K \cdot K \cdot K)$.

It remains to show that $(C,Q) \in \mathrm{BC}'(\mathcal{O})$ if and only if $C' \in \mathrm{BC}(\mathcal{O})$.

Assume $(C,Q) \in \mathrm{BC}'(\mathcal{O})$ and choose an assignment $X_1 \dots, X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ such that in the circuit $C(X_1,\dots,X_n)$, the set g_C is balanced and $K = \{1\}$ for all $K \in Q$. Let us consider the circuits $C(X_1,\dots,X_n)$ and $C'(X_1,\dots,X_n)$. Note that except for the unique assigned input gate in C' that computes $\{0\}$, there is no gate in one of the two circuits that computes a set containing 0. In particular, $g' = g$ for each unassigned input g. Hence all nodes in the circuit C' that are also contained in C compute the same set in both circuits $C(X_1,\dots,X_n)$ and $C'(X_1,\dots,X_n)$. Therefore, $g_{C'} = g_C$ is balanced.

Conversely, let C' be balanced under some assignment $X_1,\dots,X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$. As steps 1a and 1b "replace" each unassigned input g with a node computing $g - \{0\}$, we may assume that $X_1,\dots,X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ and thus every gate in $C'(X_1,\dots,X_n)$ computes a set in $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$. Let us consider the circuits $C(X_1,\dots,X_n)$ and $C'(X_1,\dots,X_n)$. As $g = g'$ for all unassigned inputs

g in $C'(X_1, \ldots, X_n)$, all nodes contained in the circuit $C(X_1, \ldots, X_n)$ compute the same set in both circuits $C(X_1, \ldots, X_n)$ and $C'(X_1, \ldots, X_n)$, which is the reason why in the following the set computed by some gate g may be denoted by g without specifying which of the two circuits we refer to. Recall that $g_{C'} = g_C \cdot \prod_{K \in Q} K^3$ is balanced. Therefore, if $K = \{1\}$ for all $K \in Q$, then $g_C = g_{C'}$ is balanced (in both circuits), and hence $(C, Q) \in \mathrm{BC}'(\mathcal{O})$.

For a contradiction, assume $K \neq \{1\}$ for some $K \in Q$. If $K = \emptyset$, then $g_{C'} = \emptyset$, which contradicts the fact that $g_{C'}$ is balanced. Hence $K \neq \emptyset$ and as furthermore $0 \notin K$ and $K \neq \{1\}$, we have $\max(K) \geq 2$. By construction, $g_{C'} = M \cdot K \cdot K \cdot K$ for some $M \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$. Thus the second statement of Lemma 4.1.3 yields that $g_{C'}$ is unbalanced, a contradiction. □

Let $\mathcal{O} = \{-, \cdot\}$ for the remainder of this section unless stated differently. In order to prove Theorem 4.2.1, we introduce certain $\mathcal{O}$-circuits which will be used extensively as components of circuits expressing Diophantine equations. Recall that pairs (C, Q) for an $\mathcal{O}$-circuit C and a subset Q of the nodes of C are also called $\mathcal{O}$-circuits (cf. Definition 4.2.2).

Lemma 4.2.4 *Let $n \in \mathbb{N}^+$. For every n-element set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{P}$ there is an $\mathcal{O}$-circuit (C_P, Q_P) that contains gates $g_{1,P}, \ldots, g_{n,P}$ satisfying the following properties:*

1. *For every assignment with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$*

$$\big(\forall_{K \in Q_P}\ K = \{1\}\big) \quad \Rightarrow \quad \exists_{m \in \mathbb{N}}\ \forall_{i=1,\ldots,n}\ g_{i,P} = \{1, p_i, \ldots, p_i^m\}.$$

2. *For each $m \in \mathbb{N}$ there is an assignment with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ under which $g_{i,P} = \{1, p_i, \ldots, p_i^m\}$ for all i and $K = \{1\}$ for all $K \in Q_P$.*

Proof We construct (C_P, Q_P) as follows:

- For each $p \in P$ insert an input gate X_p and gates $h_p = X_p - (X_p \cdot \{p\})$ and $h'_p = (\{1, p\} \cdot X_p) - (X_p - \{1\})$. For all $p \in P$ put h_p into Q_P.
- Similarly, for each $i \in \{1, \ldots, n-1\}$ and $k = p_i \cdot p_{i+1}$ insert an input gate X_k and gates $h_k = X_k - (X_k \cdot \{k\})$ and $h'_k = (\{1, k\} \cdot X_k) - (X_k - \{1\})$. Add all nodes h_k into Q_P.
- For each $i \in \{1, \ldots, n-1\}$ and $k = p_i \cdot p_{i+1}$ add a node $\gamma_k = h'_k - \big((h'_{p_i} \cdot h'_{p_{i+1}}) - \{1\}\big)$ and insert it into Q_P.
- Denote $g_{i,P} = X_{p_i}$.

For formal reasons choose an arbitrary gate as output gate. However, the circuits constructed in this proof will only be used as components of other circuits which have output gates outside of these components. Therefore, it is irrelevant which node we choose as output node. We now argue that the two assertions of the present lemma are satisfied.

1. Choose an arbitrary assignment with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ and consider the circuit under this assignment. Then no gate in the circuit computes a set containing 0. Assume $K = \{1\}$ for all $K \in Q_P$. Then in particular, for $\alpha \in \{p_1, \ldots, p_n, p_1 \cdot p_2, p_2 \cdot p_3, \ldots, p_{n-1} \cdot p_n\}$,

$$X_\alpha - X_\alpha \cdot \{\alpha\} = \{1\} \tag{4.1}$$

and $1 \in X_\alpha$.

Assume there is some $\beta \in X_\alpha$ such that β is not a power of α. Let β be minimal with that property. As $\beta \neq 0$, there are $\kappa \in \mathbb{N}$ and $\alpha' \geq 2$ with $\beta = \alpha^\kappa \cdot \alpha'$ and $\alpha \nmid \alpha'$. Due to (4.1) we obtain $\beta \in X_\alpha \cdot \{\alpha\}$ and thus $\alpha \mid \beta$. If $\kappa = 0$, we have $\beta = \alpha'$ and thus $\alpha \mid \alpha'$, a contradiction.

If $\kappa > 0$, then we obtain from $\beta \in X_\alpha \cdot \{\alpha\}$ that $\alpha^{\kappa-1} \cdot \alpha' \in X_\alpha$, which is a contradiction to the choice of β. Thus X_α only contains powers of α.

Now choose $\kappa \in \mathbb{N}$ with $\alpha^\kappa \in X_\alpha$. Then $\alpha^\kappa = 1$ or —in case $\kappa > 0$— due to (4.1) we have $\alpha^\kappa \in X_\alpha \cdot \{\alpha\}$ and thus $\alpha^{\kappa-1} \in X_\alpha$. Hence each X_α is of the form $\{1, \alpha, \ldots, \alpha^{m_\alpha}\}$ for some $m_\alpha \in \mathbb{N}$. As a consequence, $h'_\alpha = \{1, \alpha^{m_\alpha+1}\}$.

Let $k = p_i \cdot p_{i+1}$ for some i. As $\gamma_k = \{1\}$, we obtain

$$k^{m_k+1} = p_i^{m_k+1} \cdot p_{i+1}^{m_k+1} \in \left(\{1, p_i^{m_{p_i}+1}\} \cdot \{1, p_{i+1}^{m_{p_{i+1}}+1}\}\right),$$

which yields $m_k = m_{p_i} = m_{p_{i+1}}$. Thus there exists m such that for each $i \in \{1, \ldots, n\}$ it holds $g_{i,P} = \{1, p_i, \ldots, p_i^m\}$.

2. Let $m \in \mathbb{N}$ and choose the assignment with $X_\alpha = \{1, \alpha, \ldots, \alpha^m\}$ for $\alpha \in \{p_1, \ldots, p_n, p_1 \cdot p_2, p_2 \cdot p_3, \ldots, p_{n-1} \cdot p_n\}$. Consider the circuit under this assignment and observe that $h_\alpha = \{1\}$ and $h'_\alpha = \{1, \alpha^{m+1}\}$ for all α. Consequently, for $k = p_i p_{i+1}$ with $1 \leq i \leq n-1$ it holds

$$\gamma_k = \{1, p_i^{m+1} \cdot p_{i+1}^{m+1}\} - \{p_i^{m+1}, p_{i+1}^{m+1}, p_i^{m+1} \cdot p_{i+1}^{m+1}\} = \{1\},$$

which proves the second statement. □

Building upon this construction we extend these circuits and obtain the following lemma.

Lemma 4.2.5 *Let $n \in \mathbb{N}^+$. For every n-element set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{P}$ there is an $\mathcal{O}$-circuit (C_P, Q_P) with gates $g_P^0, g_P^1, \ldots, g_P^n$ satisfying the following properties:*

1. *For each assignment with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ it holds*

$$\forall_{K \in Q_P}\, K = \{1\} \quad \Rightarrow \quad \text{for all } i = 0, \ldots, n \text{ it holds that } 1 \in g_P^i,\ |g_P^i| = |g_P^1|^i, \text{ and all prime divisors of numbers in } g_P^i \text{ are in } P.$$

2. *For each $m \in \mathbb{N}^+$ there is an assignment with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ under which for all $i = 1, \ldots, m$, (i) $|g_P^i| = m^i$, (ii) $1 \in g_P^i$, (iii) all prime divisors of numbers in g_P^i are in P, and (iv) $K = \{1\}$ for all $K \in Q_P$.*

Proof Let (C'_P, Q'_P) and $g_{1,P}, \ldots, g_{n,P}$ satisfy the properties specified in Lemma 4.2.4. Add nodes $g_P^0, g_P^1, \ldots, g_P^n$ such that $g_P^0 = \{1\}$ and $g_P^i = g_P^{i-1} \cdot g_{i,P}$ for $i = 1, \ldots, n$. We call the circuit obtained this way (C_P, Q_P) (for formal reasons an arbitrary gate can be chosen as output gate; however, the circuits constructed in this proof will only be used as components of other circuits with output gates outside of these components).

Consider the circuit C_P under an arbitrary assignment under which there exists $m \in \mathbb{N}^+$ such that for all $i = 1, \ldots, n$ it holds $g_{i,P} = \{1, p_i, \ldots, p_i^{m-1}\}$. For $g_P^0 = \{1\}$, trivially $1 \in g_P^0$, $|g_P^0| = |g_P^1|^0$, and the numbers in g_P^0 do not have any prime divisors at all. A simple induction shows that for all $i = 1, \ldots, n$, (i) all prime divisors of numbers in g_P^i are in $\{p_1, \ldots, p_i\}$, (ii) $|g_P^i| = |g_P^{i-1}| \cdot |g_{i,P}| = |g_P^1|^i = m^i$, and (iii) $1 \in g_P^i$.

Thus the first statement follows from $\forall_{K \in Q_P}\, K = \{1\}$ and Lemma 4.2.4.1 and the second statement is implied by Lemma 4.2.4.2. □

By Lemma 4.2.3, it suffices to show the reduction $\mathcal{DE} \leq_{\text{m}} \text{BC}'(-,\cdot)$ in order to prove Theorem 4.2.1. Instead of showing this reduction directly we define an intermediate problem, the cardinality circuit problem

$$\mathcal{CC} = \{(C, Q, s, t) \mid C = (V, E, g_C, \alpha, \beta) \text{ is a } \{-,\cdot\}\text{-circuit, } Q \subseteq V,\ s, t \in V, \text{ and there exists an assignment with values from } \mathcal{P}_{\text{fin}}(\mathbb{N}^+) \text{ under which (i) } |s| = |t| \text{ and (ii) } K = \{1\} \text{ for all } K \in Q\}.$$

For the sake of simplicity, tuples (C, Q, s, t) are also called $\{-, \cdot\}$-circuits. In the following we do not only reduce $\mathcal{DE}$ to $\mathcal{CC}$ but we reduce it in a way such that the reduction function solely maps to circuits with some additional properties. The class $\mathcal{SC}$ of these circuits is defined as follows:

$$\mathcal{SC} = \{(C, Q, s, t) \mid C = (V, E, g_C, \alpha, \beta) \text{ is a } \{-, \cdot\}\text{-circuit, } Q \subseteq V,\ s, t \in V, \text{ under all assignments with values from } \mathcal{P}_{\text{fin}}(\mathbb{N}^+) \text{ it holds: if } \forall_{K \in Q}\ K = \{1\}, \text{ then}$$

1. $|s| \geq |t|$,
2. $1 \in s \cap t$, and
3. all prime divisors of numbers in $s \cup t$ are greater than 3$\}$.

In order to prove Theorem 4.2.1, we show

(i) $(\mathcal{DE}, \overline{\mathcal{DE}}) \leq_{\text{m}} (\mathcal{CC} \cap \mathcal{SC}, \overline{\mathcal{CC}} \cap \mathcal{SC})$ and

(ii) $(\mathcal{CC} \cap \mathcal{SC}, \overline{\mathcal{CC}} \cap \mathcal{SC}) \leq_{\text{m}} (\text{BC}'(-, \cdot), \overline{\text{BC}'(-, \cdot)})$.

The function composition of the two reduction functions yields a reduction $\mathcal{DE} \leq_{\text{m}} \text{BC}'(-, \cdot)$.

Lemma 4.2.6 $(\mathcal{DE}, \overline{\mathcal{DE}}) \leq_{\text{m}} (\mathcal{CC} \cap \mathcal{SC}, \overline{\mathcal{CC}} \cap \mathcal{SC})$.

Proof We present an algorithm that accomplishes the required reduction.

Let $q(x_1, \ldots, x_n)$ and $q'(x_1, \ldots, x_n)$ be multivariate polynomials. Then for all $y_1, \ldots, y_n \in \mathbb{N}^+$, it holds

$$q(y_1, \ldots, y_n) = q'(y_1, \ldots, y_n) \Leftrightarrow q(y_1, \ldots, y_n)^2 + q'(y_1, \ldots, y_n)^2 = 2 \cdot q(y_1, \ldots, y_n) \cdot q'(y_1, \ldots, y_n)$$

and

$$q(y_1, \ldots, y_n)^2 + q'(y_1, \ldots, y_n)^2 \geq 2 \cdot q(y_1, \ldots, y_n) \cdot q'(y_1, \ldots, y_n).$$

Therefore, we may assume that the algorithm is given multivariate polynomials $q(x_1, \ldots, x_n)$ and $q'(x_1, \ldots, x_n)$ as inputs which satisfy

$$\forall_{y_1, \ldots, y_n \in \mathbb{N}^+}\ q(y_1, \ldots, y_n) \geq q'(y_1, \ldots, y_n). \tag{4.2}$$

Let $m, m', a_1, \ldots, a_m, a'_1, \ldots, a'_{m'} \in \mathbb{N}^+$ and $d_{i,j}, d'_{\kappa,j} \in \mathbb{N}$ for $i = 1, \ldots, m$, $\kappa = 1, \ldots, m'$, and $j = 1, \ldots, n$ such that

$$q(x_1, \ldots, x_n) = \sum_{i=1}^{m} a_i \cdot \prod_{j=1}^{n} x_j^{d_{i,j}} \quad \text{and} \quad q'(x_1, \ldots, x_n) = \sum_{i=1}^{m'} a'_i \prod_{j=1}^{n} x_j^{d'_{i,j}}.$$

Moreover, for $j = 1, \ldots, n$ define $e_j = \max(\{d_{1,j}, \ldots, d_{m,j}, d'_{1,j}, \ldots, d'_{m',j}\})$, i.e., e_j denotes the maximal exponent of the variable x_j occurring in a monomial of q or q'.

We now successively build up the output circuit $(C, Q, z_q, z_{q'})$ and initially let $C = Q = \emptyset$ (the nodes z_q and $z_{q'}$ will be introduced in the following algorithm).

1. For each variable x_j, choose an e_j-element set $P_j = \{p_{j,1}, \ldots, p_{j,e_j}\} \subseteq \mathbb{P}^{>3}$ such that $P_j \cap P_{j'} = \emptyset$ for $j \neq j'$, insert a circuit (C_{P_j}, Q_{P_j}) having the properties specified in Lemma 4.2.5 into C, and insert the nodes of Q_{P_j} into Q.

   We make use of the notation of Lemma 4.2.5, in particular, we refer to the nodes $g^0_{P_j}, \ldots, g^{e_j}_{P_j}$.

   Intuition: the cardinality of $g^i_{P_j}$ is supposed to describe the value of x_j^i.

2. (a) Choose a prime $\pi > 3$ not used before and insert gates such that for all $i = 1, \dots, m$ there is a gate $h_i = \{1, \pi, \dots, \pi^{a_i - 1}\} \cdot \prod_{j=1}^{n} g_{P_j}^{d_{i,j}}$.
Intuition: the cardinality of h_i describes the value of the i-th monomial of q.

   (b) For $i = 1, \dots, m$ choose a prime $\pi_i > 3$ not used before and insert a node $h'_i = (\{1, \pi_i\} \cdot h_i) - (h_i - \{1\})$.
Intuition: as addition is supposed to be expressed by union, we need to make sure that the sets standing for distinct monomials are (almost) disjoint. Still, for a technical reason we have to keep 1 in each set. So the idea is to let h'_i consist of 1 and a copy of h_i multiplied with an additional prime factor.

   (c) For $i = 1, \dots, m$ add an unassigned input node z_q. Finally, add $m + 1$ nodes $z_q - \left(\bigcup_{i=1}^{m} h'_i - \{1\}\right)$ and $h'_i - (z_q - \{1\})$ for $i = 1, \dots, m$ and insert all these nodes into Q.
Intuition: z_q describes the value of the polynomial $q + 1$ as it is the union of the h'_i (for $i = 1, \dots, m$).

3. Do the same as in step 2 but for q'. In particular, an unassigned input node $z_{q'}$ is added.

4. Return $(C, Q, z_q, z_{q'})$.

First, observe that the function $(q, q') \mapsto (C, Q, z_q, z_{q'})$ is computable. In order to show

$$(q, q') \in \mathcal{DE} \Rightarrow (C, Q, z_q, z_{q'}) \in \mathcal{CC} \cap \mathcal{SC} \quad \text{and} \quad (q, q') \notin \mathcal{DE} \Rightarrow (C, Q, z_q, z_{q'}) \in \overline{\mathcal{CC}} \cap \mathcal{SC},$$

we make the following central observations.

Claim 4.2.7 *1. For all $y_1, \dots, y_n \in \mathbb{N}^+$ there is an assignment of the unassigned inputs of C with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ under which $\forall_{K \in Q}\ K = \{1\}$ and $\forall_{j=1,\dots,n}\ |g^1_{P_j}| = y_j$.*

2. If under some assignment with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ it holds $\forall_{K \in Q}\ K = \{1\}$, then under this assignment (i) $1 \in z_q \cap z_{q'} \cap \bigcap_{j=1}^{n} g^1_{P_j}$, (ii) $|z_q| = 1 + q(|g^1_{P_1}|, \dots, |g^1_{P_n}|)$, (iii) $|z_{q'}| = 1 + q'(|g^1_{P_1}|, \dots, |g^1_{P_n}|)$, and (iv) all prime divisors of numbers in $z_q \cup z_{q'}$ are > 3.

Proof 1. According to Lemma 4.2.5.2 the unassigned inputs of the circuits (C_{P_j}, Q_{P_j}) can be assigned with sets from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ such that

- $K = \{1\}$ for all $K \in Q_{P_j}$ and
- for each $j \in \{1, \dots, n\}$ and each $i \in \{0, \dots, e_j\}$ it holds $|g^i_{P_j}| = y^i_j$ and $1 \in g^i_{P_j}$.

Note that although we have not defined the assignment completely yet, the sets computed by the gates $h'_1, \dots, h'_m$ are already fixed. Now extend this assignment such that z_q is assigned with the set $\bigcup_{i=1}^{m} h'_i$ and observe that under the (now completely specified) assignment it holds $\forall_{i=1,\dots,m}\ 1 \in h_i$ and thus $\forall_{i=1,\dots,m}\ 1 \in h'_i$.

We have already seen that $K = \{1\}$ for all such nodes K which the algorithm adds into Q in step 1. Moreover, by the assignment of z_q and $\forall_{i=1,\dots,m}\ 1 \in h'_i$, all nodes added into Q in step 2c compute $\{1\}$. The analogous holds for the nodes inserted into Q in step 3. This completes the proof of the first statement.

2. We focus on the statements for the gate z_q. Consider the circuit under an assignment with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ such that $K = \{1\}$ for all $K \in Q$. Then $1 \in z_q$,

$$\forall_{i=1,\dots,m}\ 1 \in h'_i, \tag{4.3}$$

and

$$z_q = \bigcup_{i=1}^{m} h'_i \tag{4.4}$$

as $z_q - \left(\bigcup_{i=1}^{m} h'_i - \{1\}\right) = \{1\}$ and $h'_i - (z_q - \{1\}) = \{1\}$ for $i = 1, \ldots, m$.

Moreover, according to Lemma 4.2.5.1 for each $j = 1, \ldots, n$ and every $i = 0, \ldots, e_j$ it holds that $1 \in g^i_{P_j}$, $|g^i_{P_j}| = |g^1_{P_j}|^i$, and all prime divisors of numbers in $g^i_{P_j}$ are in P_j. As $P_j \cap P_{j'} = \emptyset$ for $j \neq j'$ and the prime $\pi > 3$ chosen in step 2a is not contained in the set $\bigcup_{j=1}^{n} P_j$, we obtain

$$\forall_{i=1,\ldots,m}\ |h_i| = a_i \cdot \prod_{j=1}^{n} |g^1_{P_j}|^{d_{i,j}} \tag{4.5}$$

and that for $i = 1, \ldots, m$ all prime divisors of numbers in h_i are in $\{\pi\} \cup \bigcup_{j=1}^{n} P_j$. Hence by (4.3) and because all primes π_i chosen in step 2b are not contained in $\{\pi\} \cup \bigcup_{j=1}^{n} P_j$, we obtain

$$\forall_{i=1,\ldots,m}\ h'_i = \{\pi_i\} \cdot h_i \cup \{1\}. \tag{4.6}$$

By that, $\forall_{i \neq i'}\ \pi_i \neq \pi_{i'}$, and as the primes π_i are not contained in $\{\pi\} \cup \bigcup_{j=1}^{n} P_j$, it holds

$$\forall_{i \neq i'}\ h'_i \cap h'_{i'} = \{1\}. \tag{4.7}$$

Hence

$$\begin{aligned} |z_q| &\overset{(4.4)}{=} \left|\bigcup_{i=1}^{m} h'_i\right| \overset{(4.7)}{=} 1 + \sum_{i=1}^{m} (|h'_i| - 1) \overset{(4.6)}{=} 1 + \sum_{i=1}^{m} |h_i| \overset{(4.5)}{=} 1 + \sum_{i=1}^{m} a_i \cdot \prod_{j=1}^{n} |g^1_{P_j}|^{d_{i,j}} \\ &= 1 + q(|g^1_{P_1}|, \ldots, |g^1_{P_n}|). \end{aligned}$$

We have seen above that for $i = 1, \ldots, m$ all prime divisors of numbers in h_i are in $\{\pi\} \cup \bigcup_{j=1}^{n} P_j \subseteq \mathbb{P}^{>3}$. Together with $\forall_{i=1,\ldots,m}\ \pi_i > 3$, (4.6) and (4.4) this yields that all prime divisors of numbers in z_q are > 3 as well.

Now we have proven the statements for the gate z_q. The proof of the corresponding statements for $z_{q'}$ is analogous. This finishes the proof of Claim 4.2.7. □

Claim 4.2.8 *1. If $(q, q') \in \mathcal{DE}$, then $(C, Q, z_q, z_{q'}) \in \mathcal{CC} \cap \mathcal{SC}$.*

2. If $(q, q') \notin \mathcal{DE}$, then $(C, Q, z_q, z_{q'}) \in \overline{\mathcal{CC}} \cap \mathcal{SC}$.

Proof Observe that $(C, Q, z_q, z_{q'}) \in \mathcal{SC}$ by (4.2) and Claim 4.2.7.2. Hence it remains to show (i) $(q, q') \in \mathcal{DE} \Rightarrow (C, Q, z_q, z_{q'}) \in \mathcal{CC}$ and (ii) $(q, q') \notin \mathcal{DE} \Rightarrow (C, Q, z_q, z_{q'}) \in \overline{\mathcal{CC}}$.

Let us argue for (i). Assume $(q, q') \in \mathcal{DE}$. Then there are $y_1, \ldots, y_n \in \mathbb{N}^+$ with $q(y_1, \ldots, y_n) = q'(y_1, \ldots, y_n)$. Claim 4.2.7.1 yields that there is an assignment of the unassigned inputs of C with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ under which $\forall_{K \in Q}\ K = \{1\}$ and $\forall_{j=1,\ldots,n}\ |g^1_{P_j}| = y_j$. By Claim 4.2.7.2, under this assignment

$$|z_q| = 1 + q(|g^1_{P_1}|, \ldots, |g^1_{P_n}|) = 1 + q(y_1, \ldots, y_n) = 1 + q'(y_1, \ldots, y_n) = 1 + q'(|g^1_{P_1}|, \ldots, |g^1_{P_n}|) = |z_{q'}|,$$

which shows $(C, Q, z_q, z_{q'}) \in \mathcal{CC}$.

Now we argue for (ii) and prove the contraposition. Assume $(C, Q, z_q, z_{q'}) \in \mathcal{CC}$. Then there is an assignment of the unassigned inputs of C with values from $\mathcal{P}_{\text{fin}}(\mathbb{N}^+)$ under which $\forall_{K \in Q}\ K = \{1\}$ and $|z_q| = |z_{q'}|$. By Claim 4.2.7.2, under this assignment

$$q(|g^1_{P_1}|, \ldots, |g^1_{P_n}|) = |z_q| - 1 = |z_{q'}| - 1 = q'(|g^1_{P_1}|, \ldots, |g^1_{P_n}|)$$

and $1 \in g^1_{P_j}$ for all $j = 1, \ldots, n$, which yields $|g^1_{P_j}| \in \mathbb{N}^+$ for all $j = 1, \ldots, n$. Thus $(q, q') \in \mathcal{DE}$, which finishes the proof of Claim 4.2.8. □

This finishes the proof of Lemma 4.2.6. □

Let us now prove the second reduction.

Lemma 4.2.9 $(\mathcal{CC} \cap \mathcal{SC}, \overline{\mathcal{CC}} \cap \mathcal{SC}) \leq_{\mathrm{m}} (\mathrm{BC}'(-,\cdot), \overline{\mathrm{BC}'(-,\cdot)})$.

Proof The following algorithm computes the reduction function. The comments in italics are supposed to give intuition.

1. Let a circuit (C, Q, s, t) be given. We construct a circuit (C', Q') by successively updating the given circuit. So we start with $C' = C$ and $Q' = Q$ and use C' and Q' as program variables.

2. Add new unassigned input gates X and X'. Insert the following nodes into C' and add them into Q':

$$\{1,2\} \cdot s - (X - \{1\}), \tag{4.8}$$
$$\{1,2\} \cdot t - (X - \{1\}), \tag{4.9}$$
$$\{1,2\} \cdot (X - s) - \big((X' \cup (X - s)) - \{1\}\big), \tag{4.10}$$
$$X' - \{2\} \cdot (X - s). \tag{4.11}$$

   The basic idea is as follows: X is supposed to be an interval with $X \supseteq s \cup t$ and $\max(X) > \max(s \cup t)$ and X' is supposed to be the set $\{1\} \cup \{2\} \cdot (X - s)$, which implies $X' \cap t = \{1\}$, since t does not contain any even numbers. Then as $|s| \geq |t|$, the set $X' \cup t = \{2\} \cdot (X - s) \cup t$ contains $\max(X')/2 + |t| - |s| \leq \max(X')/2$ elements and thus is subbalanced. But if $|s| = |t|$, then $X' \cup t$ is almost balanced. Adding the element $\max(X')+1$ would make the set balanced. Such an element is generated in the next step.

3. Let $p_1 = 2$ and $p_2 = 3$. Add a circuit $(C_{\{p_1,p_2\}}, Q_{\{p_1,p_2\}})$ according to Lemma 4.2.4. Insert all nodes of $Q_{\{p_1,p_2\}}$ into Q'. Add a node $g = \big(g_{2,\{p_1,p_2\}} \cdot \{1,3\}\big) - (g_{2,\{p_1,p_2\}} - \{1\})$.

   Now g consists of 1 and some power of 3. If X, X', and g have been chosen such that $X = \{1, 2, \ldots, {}^{(\max(g)-1)}/_2\} \supseteq \{1,2\} \cdot (s \cup t)$ and $X' = \{1\} \cup \{2\} \cdot (X - s)$ and moreover, it holds $|s| = |t|$, then $X' \cup t \cup g$ is balanced. Otherwise, the set is subbalanced or has an even maximum. So the set $X' \cup t \cup g$ has the desired properties and in the next step we introduce a new unassigned output node which is supposed to compute this set.

4. Add a new unassigned input node O. Then introduce the following four nodes into C' also add them to Q':

$$O - \Big(\big(X' \cup t \cup g\big) - \{1\}\Big), \tag{4.12}$$
$$X' - (O - \{1\}), \tag{4.13}$$
$$t - (O - \{1\}), \tag{4.14}$$
$$g - (O - \{1\}). \tag{4.15}$$

5. Modify C' such that O is the output node of C'. Return (C', Q').

In order to finish the proof of Lemma 4.2.9, it remains to prove $(C, Q, s, t) \in \mathcal{CC} \cap \mathcal{SC} \Leftrightarrow (C', Q') \in \mathrm{BC}'(-,\cdot)$. We split this statement into two claims.

Claim 4.2.10 *If* $(C, Q, s, t) \in \mathcal{CC} \cap \mathcal{SC}$, *then* $(C', Q') \in \mathrm{BC}'(-, \cdot)$.

Proof Let $(C, Q, s, t) \in \mathcal{CC} \cap \mathcal{SC}$. As $(C, Q, s, t) \in \mathcal{CC}$, there is some assignment with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ under which $|s| = |t|$ and $K = \{1\}$ for all $K \in Q$. Let us consider C under this assignment. $(C, Q, s, t) \in \mathcal{SC}$ yields that $1 \in s \cap t$ and all prime divisors of numbers in $s \cup t$ are > 3. Now consider the circuit C' under the same assignment and extend this assignment in the following way such that the unassigned inputs of $C_{\{p_1,p_2\}}$, X, X', and O are assigned with an element of $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$:

- Assign the unassigned inputs of $C_{\{p_1,p_2\}}$ with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ such that (i) $g_{2,\{p_1,p_2\}} = \{1, 3, 3^2, \ldots, 3^{m-1}\}$ for $m \in \mathbb{N}^+$ minimal with $4 \cdot \max(s \cup t) < 3^m$ and (ii) $K = \{1\}$ for all $K \in Q_{\{p_1,p_2\}}$. Such an assignment exists by Lemma 4.2.4.2.
- $X = \{x \mid 1 \leq x \leq {}^{(3^m-1)}/_2\}$.
- $X' = \{1\} \cup \{2\} \cdot (X - s)$.
- $O = X' \cup t \cup g = \big(\{2\} \cdot (X - s)\big) \cup t \cup \{3^m\}$. For the second equation note $g = \{1, 3^m\}$ and recall that $1 \in t$ was observed above.

Now the assignment is defined for all unassigned inputs of C' and for the remainder of the proof of the present claim we consider the circuit under this assignment.

In order to see that $K = \{1\}$ for all $K \in Q'$, it suffices to consider the nodes added into Q' in steps 2 and 4 as the nodes added into Q' in step 3 compute $\{1\}$ by the choice of the unassigned inputs of $C_{\{p_1,p_2\}}$. So we have to prove that the nodes in (4.8), (4.9), (4.10), (4.11), (4.12), (4.13), (4.14), and (4.15) all compute $\{1\}$.

By the choice of m and by $1 \in s \cap t$, it holds $\max(X) = {}^{(3^m-1)}/_2 \geq 2 \cdot \max(s \cup t) > \max(s \cup t)$, which shows that the nodes defined in (4.8) and (4.9) compute $\{1\}$,

$$\max(X') = 2 \cdot \max(X), \tag{4.16}$$

and

$$\max(O) = \max(g) = 3^m. \tag{4.17}$$

The choice of X' immediately implies that the node defined in (4.11) computes $\{1\}$. It holds $\{1, 2\} \cdot (X - s) - \big((X' \cup (X - s)) - \{1\}\big) = \{1, 2\} \cdot (X - s) - \big((\{1, 2\} \cdot (X - s)) - \{1\}\big) = \{1\}$ and hence the node defined in (4.10) also computes $\{1\}$.

The nodes defined in (4.12), (4.13), (4.14), and (4.15) compute $\{1\}$ by the choice of the assignment (in particular, recall $1 \in O \cap X' \cap t \cap g$). Thus $K = \{1\}$ for all $K \in Q'$.

Recall that t contains only such numbers whose prime divisors are > 3 and that $\max(s) \leq \max(X)$. Hence the three sets $\{2\} \cdot (X - s)$, t, and $\{3^m\}$ are pairwise disjoint and it holds $s \subseteq X$. Thus

$$|O| = |\{2\} \cdot (X - s)| + |t| + 1 = \max(X) - |s| + |t| + 1 = \max(X) + 1 \overset{(4.16)}{=} \frac{\max(X')}{2} + 1 \overset{(4.17)}{=} \frac{\max(O) + 1}{2}$$

So O is balanced. Together with the above observation that $K = \{1\}$ for all $K \in Q'$ this shows $(C', Q') \in \mathrm{BC}'(-, \cdot)$, which completes the proof of Claim 4.2.10. □

Claim 4.2.11 *If* $(C, Q, s, t) \in \overline{\mathcal{CC}} \cap \mathcal{SC}$, *then* $(C', Q') \in \overline{\mathrm{BC}'(-, \cdot)}$.

Proof For a contradiction, assume that $(C, Q, s, t) \in \overline{\mathcal{CC}} \cap \mathcal{SC}$ and $(C', Q') \in \mathrm{BC}'(-,\cdot)$. Then there is an assignment of the unassigned inputs of C' with values from $\mathcal{P}_{\mathrm{fin}}(\mathbb{N}^+)$ under which O is balanced and all $K \in Q'$ satisfy $K = \{1\}$. For the remainder of the proof let us consider C' under this assignment and C under the restriction of this assignment to the unassigned inputs of C. As C' can be considered as an expanded version of C, each gate in C computes the same set in both circuits C and C'. Hence for each gate g that is in one of the two circuits C and C', we may write g for the set computed by g without specifying which of the two circuits we refer to.

Since the nodes defined in (4.8) and (4.9) compute $\{1\}$, we obtain $1 \in s$ and $X \supseteq \big(\{1,2\} \cdot s \cup \{1,2\} \cdot t\big) - \{1\}$. As the node defined in (4.10) also computes $\{1\}$, it even holds $X \supseteq \{1,2\} \cdot s \cup \{1,2\} \cdot t$. In particular,

$$s \cup t \subseteq X \tag{4.18}$$

and $1 \leq \max(s) < 2 \cdot \max(s) \leq \max(X)$. Due to that inequality and $\{1,2\} \cdot (X - s) - \big((X' \cup (X - s)) - \{1\}\big) = \{1\}$ (cf. (4.10)) it holds $2 \cdot \max(X) = 2 \cdot \max(X - s) \in X'$. Since the node defined in (4.11) computes $\{1\}$, we obtain $X' \subseteq \{1\} \cup \{2\} \cdot (X - s)$. In particular,

$$\max(X') = 2 \cdot \max(X). \tag{4.19}$$

The fact that the nodes defined in (4.12), (4.13), (4.14), and (4.15) all compute $\{1\}$ implies $1 \in O \cap X' \cap t \cap g$ and $O = X' \cup t \cup g$. Furthermore, as by construction $Q_{\{p_1,p_2\}} \subseteq Q'$ and thus $K = \{1\}$ for all $K \in Q_{\{p_1,p_2\}}$, Lemma 4.2.4.1 yields that $g_{2,\{p_1,p_2\}} = \{1, 3, 3^2, \ldots, 3^{m-1}\}$ for some $m \in \mathbb{N}^+$ and thus $g = \{1, 3^m\}$. Because of that, $X' \subseteq \{1\} \cup \{2\} \cdot (X - s)$, and $1 \in t$, it holds that

$$O = X' \cup t \cup g \subseteq \big(\{2\} \cdot (X - s)\big) \cup t \cup \{3^m\}. \tag{4.20}$$

As O is balanced, Lemma 4.1.1 implies that $\max(O)$ is odd. As by (4.19) the maximum of X' is even,

$$\max(O) > \max(X'). \tag{4.21}$$

As $(C, Q, s, t) \in \mathcal{SC}$ and $K = \{1\}$ for all $K \in Q$, it holds $|s| \geq |t|$. Hence as —by assumption— $(C, Q, s, t) \notin \mathcal{CC}$ but $K = \{1\}$ for all $K \in Q$, it even holds

$$|s| > |t|. \tag{4.22}$$

Putting things together, we obtain

$$\begin{aligned} |O| &\overset{(4.20)}{\leq} |X - s| + |t| + 1 \overset{(4.18)}{=} \max(X) - |s| + |t| + 1 \overset{(4.22)}{<} \max(X) + 1 \\ &\overset{(4.19)}{=} \frac{\max(X') + 2}{2} \overset{(4.21)}{\leq} \frac{\max(O) + 1}{2}, \end{aligned}$$

which contradicts the assumption that O is balanced and thus completes the proof of Claim 4.2.11. □

This completes the proof of Lemma 4.2.9. □

As has been announced before, we now can easily prove Theorem 4.2.1, which is the present section's main result and states that the problem $\mathrm{BC}(-,\cdot)$ is $\leq_{\mathrm{m}}$-complete for the set RE of all computably enumerable problems.

Proof of Theorem 4.2.1 The Lemmas 4.2.6 and 4.2.9 show that

$$(\mathcal{DE}, \overline{\mathcal{DE}}) \leq_{\mathrm{m}} (\mathcal{CC} \cap \mathcal{SC}, \overline{\mathcal{CC}} \cap \mathcal{SC}) \leq_{\mathrm{m}} (\mathrm{BC}'(-,\cdot), \overline{\mathrm{BC}'(-,\cdot)}),$$

i.e., $\mathcal{DE} \leq_{\mathrm{m}} \mathrm{BC}'(-,\cdot)$. As $\mathrm{BC}'(-,\cdot) \leq_{\mathrm{m}} \mathrm{BC}(-,\cdot)$ by Lemma 4.2.3, we obtain $\mathcal{DE} \leq_{\mathrm{m}} \mathrm{BC}(-,\cdot)$. Thus $\mathrm{BC}(-,\cdot)$ is $\leq_{\mathrm{m}}$-hard for RE. Moreover, Proposition 4.1.5 yields $\mathrm{BC}(-,\cdot) \in \mathrm{RE}$. □

In the following we apply this result for circuits disallowing set difference but allowing symmetric difference. In these circuits all operations are commutative, which is the reason why such circuits get along without order functions specifying for each inner node which is the left and which is the right direct predecessor. Hence such circuits can be defined as quadruples instead of quintuples.

Corollary 4.2.12 $\mathrm{BC}(\triangle,\cap,\cdot)$ *and* $\mathrm{BC}(\triangle,\cup,\cdot)$ *are* $\leq_{\mathrm{m}}$*-complete for* RE.

Proof The two problems are in RE according to Proposition 4.1.5. Thus by Theorem 4.2.1, it suffices to show $\mathrm{BC}(-,\cdot) \leq_{\mathrm{m}} \mathrm{BC}(\triangle,\cap,\cdot)$ and $\mathrm{BC}(-,\cdot) \leq_{\mathrm{m}} \mathrm{BC}(\triangle,\cup,\cdot)$. These assertions hold as $A - B = (A \triangle B) \cap A = (A \cup B) \triangle B$ for arbitrary $A, B \subseteq \mathbb{N}$. □

This shows that even if set difference, the only non-symmetric operation, is disallowed, one arithmetic operation is still sufficient to gain undecidability.

4.3 Smaller Sets of Operations Lead to Problems in NP

Having proven $\mathrm{BC}(-,\cdot)$ to be undecidable, the obvious question is whether even a proper subset of $\{-,\cdot\}$ suffices to gain undecidability. So we investigate the computational complexity of the problems $\mathrm{BC}(\cdot)$ and $\mathrm{BC}(-)$ and for the sake of completeness, we also consider the almost trivial problem $\mathrm{BC}(\emptyset)$, whose membership in L follows from Proposition 4.1.2 as in each $\emptyset$-circuit the output node is an input node. Thus in this section we study all problems $\mathrm{BC}(\mathcal{O})$ for $\mathcal{O} \subsetneq \{-,\cdot\}$ and it turns out that they are all in NP. More precisely, each of the problems is shown to be $\leq_{\mathrm{m}}^{\log}$-complete for one of the classes L, NL, and NP.

4.3.1 Allowing Multiplication Only

This subsection's purpose is to prove the NL-completeness of $\mathrm{BC}(\cdot)$. First, we use theorems by Ford [For08a, For08b] and Koukoulopoulos [Kou14] in order to show that $A \cdot B$ for sets A and B with sufficiently large maxima is subbalanced. Second, this result is exploited by a nondeterministic logarithmic-space algorithm which accepts $\mathrm{BC}(\cdot)$. Finally, we prove the $\leq_{\mathrm{m}}^{\log}$-hardness of the problem for NL, which is a straightforward corollary of a result by McKenzie and Wagner [MW07].

In order to make use of the mentioned results by Ford and Koukoulopoulos, we introduce some notation.

Definition 4.3.1 *Define* $A\colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}$ *via*

$$(N_1, N_2) \mapsto \big|\{n_1 \cdot n_2 \mid n_1 \leq N_1, n_2 \leq N_2, n_1, n_2 \in \mathbb{N}^+\}\big|.$$

Moreover, we define a function $H\colon \mathbb{N}^+ \times \mathbb{Q} \times \mathbb{Q} \to \mathbb{N}$ *via*

$$(x, y, z) \mapsto \big|\{n \in \mathbb{N}^+ \mid n \leq x, \exists_{d \in \mathbb{N}^+, d \mid n} \text{ such that } y < d \leq z\}\big|.$$

In other words and put more illustratively, $A(N_1, N_2)$ counts the numbers in the two-dimensional multiplication table with side lengths N_1 and N_2, where the side lengths are allowed to be different.

The following theorem formulates two statements. The first is a special case of a result by Koukoulopoulos [Kou14, Theorem 1.1]. Furthermore, Koukoulopoulos [Kou14] cites

Ford [For08a, For08b] for a further theorem which contains the second statement as a special case and can be found in the aforementioned article by Koukoulopoulos [Kou14, Theorem 1.2]. Note that Koukoulopoulos and Ford indeed prove significantly stronger results than we make use of here. In particular, we only formulate the results for the two-dimensional multiplication table, whereas [Kou14] contains analogous results for the k-dimensional table for arbitrary $k \geq 2$.

Theorem 4.3.2 ([For08a, For08b, Kou14]) *There exist $k_1, k_2 \in \mathbb{N}^+$ and $3 \leq \mu_1, \mu_2 \in \mathbb{N}$ such that the following statements hold.*

1. *For all natural numbers N_1 and N_2 with $\mu_1 \leq N_1 \leq N_2$,*

$$A(N_1, N_2) \leq k_1 \cdot H(N_1 \cdot N_2, N_1/2, N_1).$$

2. *For all natural numbers $\mu_2 \leq x \in \mathbb{N}$ and $\mu_2 \leq y \in \mathbb{Q}$ with $4y^2 \leq x$,*

$$H(x, y, 2y) \leq k_2 \cdot \frac{x}{(\log y)^{1-\frac{1+\log\log 2}{\log 2}} (\log\log y)^{3/2}}.$$

The following corollary basically formulates that the product of two sets with sufficiently big maxima is subbalanced. It is not difficult to see that this corollary (and indeed even a significantly stronger result) is implied by Theorem 4.3.2.

Corollary 4.3.3 *There exists $\mu \in \mathbb{N}$ such that for all sets $A, B \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ with $\mu \leq \max(A)$ and $\mu \leq \max(B)$ the following holds:*

1. *$A \cdot B$ is subbalanced.*

2. *For all $M \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ the set $A \cdot B \cdot M$ is not balanced.*

Proof Let k_1, μ_1, k_2, and μ_2 be the numbers guaranteed by Theorem 4.3.2. Choose $\mu \geq \max(k_1, \mu_1, k_2, \mu_2)$ minimal with the property that $\log\log(\mu/2) \geq 3k_1k_2$. Let $A, B, M \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ with $\alpha := \max(A) \geq \mu$ and $\beta := \max(B) \geq \mu$.

1. Without loss of generality, $\alpha \leq \beta$. Then statement 1 of Theorem 4.3.2 implies that $|A \cdot B| \leq 1 + A(\alpha, \beta) \leq 1 + k_1 \cdot H(\alpha \cdot \beta, \alpha/2, \alpha)$. The fact that $\alpha \leq \beta$ yields $4 \cdot (\alpha/2)^2 \leq \alpha \cdot \beta$ and thus Theorem 4.3.2.2 can be applied to $H(\alpha \cdot \beta, \alpha/2, \alpha)$, which yields

$$|A \cdot B| \leq 1 + k_1k_2 \cdot \frac{\alpha \cdot \beta}{(\log \frac{\alpha}{2})^{1-\frac{1+\log\log 2}{\log 2}} (\log\log \frac{\alpha}{2})^{3/2}} \overset{\mu \leq \alpha,\ \log\log(\mu/2) \geq 3k_1k_2}{\leq} 1 + \frac{\alpha\beta}{3} < \frac{\alpha\beta}{2}.$$

2. If $M \in \{\emptyset, \{0\}\}$, then $A \cdot B \cdot M = M$ is not balanced. If $\max(M) \geq 1$, then by the first statement of the present theorem, $|A \cdot B \cdot M| \leq |(A \cup \{0\}) \cdot B \cdot (M - \{0\})| \leq |(A \cup \{0\}) \cdot B| \cdot \max(M) < {}^{(\alpha\beta \cdot \max(M))}/_2$ and thus $A \cdot B \cdot M$ is subbalanced. □

Having built the foundation, let us now turn to the problem BC$(\cdot)$, which we show to be $\leq_{\text{m}}^{\log}$-complete for NL. We prove membership and hardness in two separate theorems and start with the more complicated result.

Theorem 4.3.4 BC$(\cdot) \in$ NL.

Proof Let us start with the definition of the following auxiliary problem.

$$\Theta = \big\{(A,r) \mid A \subseteq \big(\mathcal{P}_{\text{fin}}(\mathbb{N}) - \{\emptyset,\{0\}\}\big) \times \{1,2\} \text{ finite}, \forall_{S \in \mathcal{P}_{\text{fin}}(\mathbb{N})}\ (S,1) \in A \Rightarrow (S,2) \notin A,$$
$$r \in [0,3], \Xi = \textstyle\prod_{(S,i)\in A} \prod_{j=1}^{i} S, \text{ and one of the following conditions holds:}$$

- Ξ is balanced.
- $r \geq 1$ and $\Xi \cup \{0\}$ is balanced.
- $r \geq 2$ and $\exists_{F \in \mathcal{P}_{\text{fin}}(\mathbb{N})}\ \Xi \cdot F \cdot F$ is balanced.
- $r = 3$ and $\exists_{F \in \mathcal{P}_{\text{fin}}(\mathbb{N})}\ \Xi \cdot F$ is balanced.$\big\}$.

For reasons of clarity and comprehensibility we proceed in two steps. First, we show that there is an $f \in \text{FL}^{\text{NL}}$ such that $C \in \text{BC}(\cdot) \Leftrightarrow f(C) \in \Theta$ for all $\{\cdot\}$-circuits C. Second, we show $\Theta \in \text{L}$.

Claim 4.3.5 *There is an $f \in \text{FL}^{\text{NL}}$ such that $C \in \text{BC}(\cdot) \Leftrightarrow f(C) \in \Theta$ for every $\{\cdot\}$-circuit C.*

Proof Let $N \notin \Theta$ be fixed and let f be the function computed by the following algorithm, in which we use A, r, and i_M for certain $M \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ as program variables, where A and r are global variables initialized with $\emptyset$ and 0, respectively. The algorithm is given access to the oracles $\text{GAP}_{\geq k}$ and $\text{GAP}_{=k}$ for $k \in \{1,2,3\}$, which are all in NL (cf. Section 2.3.3).[4]

1. Input: a $\{\cdot\}$-circuit $C = (V,E,g_C,\alpha,\beta)$.
2. If $\exists_{g\in V}\ \alpha(g) \in \{\emptyset,\{0\}\} \wedge (V,E,g,g_C) \in \text{GAP}_{\geq 1}$, then return N.
3. If $\exists_{g\in V}$ with $\alpha(g) = \{0,1\}$ and $(V,E,g,g_C) \in \text{GAP}_{\geq 1}$, then let $A = A \cup \{(\{0,1\},1)\}$.
4. For each $M \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ with $\max(M) \geq 2$ and $\alpha(g) = M$ for some assigned input gate g:
   (a) Let $i_M = 0$. For each assigned input gate g with $\alpha(g) = M$:
      i. If $(V,E,g,g_C) \in \text{GAP}_{=1}$ and $i_M < 3$, then $i_M = i_M + 1$.
      ii. If $(V,E,g,g_C) \in \text{GAP}_{=2}$ and $i_M < 3$, then $i_M = i_M + 2$.
      iii. If $(V,E,g,g_C) \in \text{GAP}_{\geq 3}$, then $i_M = 3$.
   (b) If $i_M \geq 3$, return N.
   (c) If $i_M \in \{1,2\}$, then let $A = A \cup \{(M,i_M)\}$.
5. For each unassigned input gate g do the following:
   (a) If $(V,E,g,g_C) \in \text{GAP}_{\geq 3}$ and $r < 1$, then $r = 1$.
   (b) If $(V,E,g,g_C) \in \text{GAP}_{=2}$ and $r < 2$, then $r = 2$.
   (c) If $(V,E,g,g_C) \in \text{GAP}_{=1}$, then $r = 3$.
6. Return (A,r)

[4] More formally, we could view the six sets as subsets of $\mathbb{N}$ and use the following set as oracle

$$\bigcup_{k=0}^{2} \big(\{6x+2k \mid x \in \text{GAP}_{\geq k+1}\} \cup \{6x+2k+1 \mid x \in \text{GAP}_{=k+1}\}\big).$$

When given access to the aforementioned oracles, the algorithm can be implemented by a deterministic logarithmic-space Turing machine. It remains to show $C \in \mathrm{BC}(\cdot) \Leftrightarrow f(C) \in \Theta$ for all $\{\cdot\}$-circuits C.

Let $C = (V, E, g_C, \alpha, \beta)$ be a $\{\cdot\}$-circuit with $k \in \mathbb{N}$ assigned input gates $g_1 < \cdots < g_k$ and $n \in \mathbb{N}$ unassigned input gates $h_1 < \cdots < h_n$.

For two arbitrary gates g and h of C let $i_{g,h}$ denote the number of paths from g to h. Recall that by definition $\prod_{i=1}^{0} A = \{1\}$ for all $A \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ and observe that by a simple induction,

$$\forall_{X_1,\ldots,X_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})} \forall_{h \in V}\ I(h; C(X_1, \ldots, X_n))) = \prod_{j=1}^{k} \prod_{j'=1}^{i_{g_j,h}} g_j \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j,h}} X_j. \tag{4.23}$$

Recall that as the set computed by an assigned input g is independent of the assignment, it holds $I(g; C(X)) = \alpha(g)$ for all $X \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})^n$ and by convention, we just write g for this set.

As an abbreviation, for an arbitrary gate g we write i_g for i_{g,g_C}. Note that each input gate not having any path to the output gate g_C neither has any influence on the above algorithm's output nor on the circuit's membership in $\mathrm{BC}(\cdot)$. Therefore, we may assume that for each input gate there exists some path to g_C, i.e.,

$$\forall_{j \in \{1,\ldots,k\}}\ i_{g_j} > 0 \quad \wedge \quad \forall_{j \in \{1,\ldots,n\}}\ i_{h_j} > 0. \tag{4.24}$$

If the algorithm returns N in step 2, then there is an assigned input gate g with $\alpha(g) \in \{\emptyset, \{0\}\}$ that is connected to the output and thus under every assignment it holds that $g_C = g \in \{\emptyset, \{0\}\}$ is not balanced. Hence $C \notin \mathrm{BC}(\cdot)$. If the algorithm returns N in step 4b, then according to step 4a there is a set $K \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ with $\max(K) \geq 2$ such that there are at least 3 paths from assigned inputs g with $\alpha(g) = K$ to the output g_C. Then by (4.23), it holds under every assignment $X \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})^n$ that $g_C = M \cdot K \cdot K \cdot K$ for some $M \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$. But then the second statement of Lemma 4.1.3 yields that g_C is not balanced and hence $C \notin \mathrm{BC}(\cdot)$. Thus, if the algorithm terminates in step 2 or step 4b, then $C \in \mathrm{BC}(\cdot) \Leftrightarrow f(C) \in \Theta$.

Consequently, for the remainder of the proof of the present claim we consider the case that the algorithm terminates in step 6. Let (A, r) denote the return value of the algorithm on input C. Observe that then A is a finite subset of $\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}) - \{\emptyset, \{0\}\}\big) \times \{1, 2\}$, $r \in [0, 3]$, and for all $S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ it holds $(S, 1) \in A \Rightarrow (S, 2) \notin A$. Hence from now on we only need to consider the four bullet points in the definition of Θ in order to determine whether $(A, r) \in \Theta$. Let

$$B = \Big\{(S, i) \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \times \mathbb{N}^+ \mid i = \sum_{j \in \{1,\ldots,k\},\, \alpha(g_j) = S} i_{g_j}\Big\}.$$

In other words, B consists of those pairs (S, i) for which there exists some assigned input gate g_j that computes S (note that by (4.24), there exists a path from g_j to the output gate) and i equals the number of paths from assigned inputs computing S to the output gate g_C. The first equation below holds by the definition of B, whereas the second equation still has to be proven.

$$\prod_{j=1}^{k} \prod_{j'=1}^{i_{g_j}} g_j = \prod_{(S,i) \in B} \prod_{i'=1}^{i} S = \prod_{(S,i) \in A} \prod_{i'=1}^{i} S. \tag{4.25}$$

Let us now argue for the second equation in (4.25). Since the algorithm does not terminate in step 2, it holds $B \cap T = \emptyset$ for $T = \{\emptyset, \{0\}\} \times \mathbb{N}$ and thus $B \cap T = A \cap T$. As the algorithm does not terminate in step 4b, it holds for all $(S, i) \in B$ that $\big(\max(S) \geq 2 \Rightarrow i \leq 2\big)$. Due to that and steps 4a and 4c, we have $B \cap T' = A \cap T'$ for $T' = \big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}) - \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}\big) \times \mathbb{N}$.

Therefore, it remains to consider the pairs in $T'' = \{\{1\},\{0,1\}\} \times \mathbb{N}$. The pairs in $\{\{1\}\} \times \mathbb{N}$ can be ignored. The algorithm either adds $(\{0,1\},1)$ or no set from T'' at all into A and step 3 ensures that $(\{0,1\},1) \in A$ if and only if there exists $i > 0$ with $(\{0,1\},i) \in B$, which finishes the proof of the second equation in (4.25).

Let us denote the set described by the three expressions in (4.25) with Ξ. By (4.23),

$$\forall_{X_1,\ldots,X_n \in \mathcal{P}_{\text{fin}}(\mathbb{N})}\ I(g_C; C(X_1,\ldots,X_n)) = \Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$$

and hence it remains to prove that

$$\left(\exists_{X_1,\ldots,X_n \in \mathcal{P}_{\text{fin}}(\mathbb{N})}\ \Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j \text{ is balanced}\right) \ \Leftrightarrow\ (A,r) \in \Theta. \tag{4.26}$$

"$\Rightarrow$": Assume there exist $X_1,\ldots,X_n \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ under which $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ is balanced. By statement 2 of Lemma 4.1.3,

$$\neg\exists_{j\in\{1,\ldots,n\}}\ \left(\max(X_j) \geq 2 \wedge i_{h_j} \geq 3\right) \tag{4.27}$$

By (4.24), it suffices to study the following cases.

- $n = 0$: In this case, Ξ is balanced and hence $(A,r) \in \Theta$.
- $n > 0 \wedge \forall_{j\in\{1,\ldots,n\}}\ i_{h_j} \geq 3$: Here, $r = 1$. (4.27) yields $\forall_{j\in\{1,\ldots,n\}}\ \max(X_j) \leq 1$ and as $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ is balanced, even $\forall_{j\in\{1,\ldots,n\}}\ \max(X_j) = 1$. Hence $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j \in \{\Xi, \Xi \cup \{0\}\}$ and thus one of the sets Ξ and $\Xi \cup \{0\}$ is balanced, which shows $(A,r) \in \Theta$.
- $n > 0 \wedge \forall_{j\in\{1,\ldots,n\}}\ i_{h_j} \geq 2 \wedge \exists_{j\in\{1,\ldots,n\}}\ i_{h_j} = 2$: In this case $r = 2$. Let $F = \prod_{j=1}^{n} X_j$. Then $F \cdot F = \prod_{j=1}^{n} \prod_{j'=1}^{2} X_j = \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$, where the second equation holds as $2 \leq i_{h_j}$ for all j and by (4.27), for all $j \in \{1,\ldots,n\}$ with $i_{h_j} > 2$ we have $\max(X_j) \leq 1$ and thus $\prod_{j'=1}^{2} X_j = \prod_{j'=1}^{i_{h_j}} X_j$. Thus $\Xi \cdot F \cdot F = \Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ is balanced and hence $(A,r) \in \Theta$.
- $n > 0 \wedge \exists_{j\in\{1,\ldots,n\}}\, i_{h_j} = 1$. Here, $r = 3$. For $F = \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ the set $\Xi \cdot F = \Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ is balanced and hence $(A,r) \in \Theta$.

"$\Leftarrow$": Assume $(A,r) \in \Theta$.

If $r = 0$, then Ξ is balanced and $n = 0$ (if $n > 0$, then due to (4.24) the algorithm returns a value $r > 0$). Thus the left-hand side of (4.26) is satisfied.

If $r = 1$, then Ξ or $\Xi \cup \{0\}$ is balanced, $n \geq 1$, and $i_{h_j} \geq 3$ for all $j = 1,\ldots,n$ (if some i_{h_j} is < 3, then due to (4.24) the algorithm returns a value $r > 1$). If Ξ is balanced, then we choose $X_j = \{1\}$ for all $j = 1,\ldots,n$. If Ξ is not balanced, then $\Xi \cup \{0\}$ is balanced and we let $X_j = \{0,1\}$ for all $j = 1,\ldots,n$. Then $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j$ is balanced.

If $r = 2$, then there exists $F \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ such that $\Xi \cdot F \cdot F$ is balanced and there is some $j \in \{1,\ldots,n\}$ with $i_{h_j} = 2$. Let $X_j = F$ and $X_{j'} = \{1\}$ for all $j' \neq j$. Then $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j = \Xi \cdot F \cdot F$ is balanced.

If $r = 3$, then there exists $F \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ such that $\Xi \cdot F$ is balanced and there is some $j \in \{1,\ldots,n\}$ with $i_{h_j} = 1$. Let $X_j = F$ and $X_{j'} = \{1\}$ for all $j' \neq j$. Then $\Xi \cdot \prod_{j=1}^{n} \prod_{j'=1}^{i_{h_j}} X_j = \Xi \cdot F$ is balanced. This shows (4.26) and finishes the proof of Claim 4.3.5. □

Claim 4.3.6 $\Theta \in \mathrm{L}$.

Proof Let μ be as specified in Corollary 4.3.3, i.e., for all $A, B, M \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ with $\mu \leq \max(A) \wedge \mu \leq \max(B)$ the set $A \cdot B \cdot M$ is not balanced. We show that the following algorithm accepts Θ.

1. Input: a pair (A, r) such that $A \subseteq \left(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}) - \{\emptyset, \{0\}\}\right) \times \{1, 2\}$ is finite, $r \in \{0, 1, 2, 3\}$, and $(S, 1) \in A \Rightarrow (S, 2) \notin A$ for all $S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ (if the pair is not of this form, reject).
2. If there are distinct $(S, i_S), (T, i_T) \in A$ with $\max(S) \geq \mu$ and $\max(T) \geq \mu$, then reject. If there is some $(S, i) \in A$ with $\max(S) \geq \mu$ and $i = 2$, then reject.
3. If $A \subseteq \mathcal{P}(\{0, \ldots, \mu\}) \times \{1, 2\}$, then accept if $(A, r) \in \Theta$ and reject otherwise.
4. *If the algorithm executes this step, then there is exactly one pair* $(S, i) \in A$ *with* $\max(S) \geq \mu$ *and for this pair* $i = 1$ *(otherwise, the algorithm would have rejected in step 2 or step 3).*
   (a) Compute the set $M = \prod_{(T,i) \in A - \{(S,1)\}} \prod_{j=1}^{i} T$.
   (b) If (i) $S \in \mathrm{BAL}_M$ or (ii) $r = 1 \wedge S \cup \{0\} \in \mathrm{BAL}_M$, then accept.
   (c) For all $F \in \mathcal{P}([0, \mu])$: accept if (i) $r = 2 \wedge S \in \mathrm{BAL}_{M \cdot F \cdot F}$ or (ii) $r = 3 \wedge S \in \mathrm{BAL}_{M \cdot F}$.
   (d) Reject.

Logarithmic space is clearly sufficient for the first two steps. In step 3, the test whether $(A, r) \in \Theta$ is only done if A is of constant size and thus logarithmic space suffices again. The fact that the sets M, $M \cup \{0\}$, $M \cdot F \cdot F$, and $M \cdot F$ in step 4 are of constant size and Proposition 4.1.2 yield that step 4 can be implemented by a deterministic logarithmic-space Turing machine.

We show that on input (A, r),if the algorithm accepts, then $(A, r) \in \Theta$, and if it rejects, then $(A, r) \notin \Theta$. We study cases depending on the step the algorithm terminates in.

Step 1: If the algorithm rejects in this step, then by definition $(A, r) \notin \Theta$. Step 2: If the algorithm rejects in this step, then $(A, r) \notin \Theta$ by Corollary 4.3.3.2. Step 3: Here the algorithm clearly accepts and rejects correctly.

Step 4: Let us assume that the algorithm terminates in this step. As argued above, due to steps 2 and 3, there is precisely one pair $(S, i) \in A$ with $\max(S) \geq \mu$ and for this pair $i = 1$. By definition of Θ, if the algorithm accepts in step 4b or in some iteration of the loop in step 4c, then $(A, r) \in \Theta$. Now study the case that the algorithm rejects in step 4d and for a contradiction, assume $(A, r) \in \Theta$. If $r < 2$, then $S \cdot M$ or $S \cdot M \cup \{0\}$ is balanced, which contradicts the fact that the algorithm does not accept in step 4b. So $r \in \{2, 3\}$ and there exists some $F \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ such that $M \cdot S \cdot F$ or $M \cdot S \cdot F \cdot F$ is balanced. However, as the algorithm does not accept in step 4c, it holds $\max(F) > \mu$, in contradiction to Corollary 4.3.3.2, which implies that if $\max(F) > \mu$, then $M \cdot S \cdot F$ and $M \cdot S \cdot F \cdot F$ are not balanced. This finishes the proof of Claim 4.3.6. □

Claim 4.3.5 and Claim 4.3.6 yield an L^{NL}-algorithm for BC$(\cdot)$. As $\mathrm{L}^{\mathrm{NL}} = \mathrm{NL}$ according to Proposition 2.3.2, the proof of Theorem 4.3.4 is complete. □

Theorem 4.3.7 BC$(\cdot)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NL.

Proof Recall that McKenzie and Wagner [MW07] prove the problem MC$(\cap)$ to be $\leq_{\mathrm{m}}^{\log}$-complete for NL. The following deterministic logarithmic-space algorithm shows MC$(\cap) \leq_{\mathrm{m}}^{\log}BC(\cdot)$.

Assume we are given as input a pair (C, b) where $C = (V, E, g_C, \alpha, \beta)$ is a $\{\cap\}$-circuit with (assigned) inputs $g_1 < \cdots < g_n$ for $n \in \mathbb{N}^+$ and $b \in \mathbb{N}$. Let $\alpha' \colon V \mapsto \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \cup \{\cdot\}$ be the total

function that maps each inner gate to $\cdot$, each input gate g with $b \notin \alpha(g)$ to $\{0\}$, and each input gate with $b \in \alpha(g)$ to $\{1\}$. Return $C' = (V, E, g_C, \alpha', \beta)$.

Note $\forall_{g \in V}\ I(g; C') \in \{\{0\}, \{1\}\}$. An induction shows $\forall_{g \in V}\ b \in I(g, C) \Leftrightarrow I(g, C') = \{1\}$. □

Corollary 4.3.8 BC($\cdot$) *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NL.

Proof The assertion follows from Theorems 4.3.4 and 4.3.7. □

4.3.2 The Problems Not Allowing Multiplication

We consider the two remaining problems and prove that BC($-$) is $\leq_{\mathrm{m}}^{\log}$-complete for NP and BC($\emptyset$) is in L (and thus trivially $\leq_{\mathrm{m}}^{\log}$-complete for L).

In order to show that BC($-$) is $\leq_{\mathrm{m}}^{\log}$-hard for NP, we first define the circuit satisfiability problem CSAT, which is the circuit version of SAT. For this purpose we define Boolean circuits. The definition is similar to the definition of integer circuits.

A *Boolean circuit* C is a quadruple (V, E, g_C, α) with the following properties:

- (V, E, g_C) is a circuit such that each node has indegree at most 2.
- $\alpha\colon V \to \{\square, \neg, \wedge, \vee\}$ is a total function.
- For each node u with indegree 0 it holds $\alpha(u) = \square$.
- For each node u with indegree 1 it holds $\alpha(u) = \neg$. Such nodes are called $\neg$-gates.
- For each node u with indegree 2 it holds $\alpha(u) \in \{\wedge, \vee\}$. Such nodes are called $\wedge$-gates and $\vee$-gates, respectively.

Note that we get along without assigned input gates, because these can be simulated by $g \wedge \neg g$ and $g \vee \neg g$ for an unassigned input g.

Let $g_1 < \cdots < g_n$ for $n \in \mathbb{N}^+$ be the input gates of a Boolean circuit $C = (V, E, g_C, \alpha)$. Inductively, we define for each assignment $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ and for each gate g which of the values in $\{0, 1\}$ the gate g computes:

$$I_a(g; C) = \begin{cases} a_i & \text{if } g = g_i \text{ for some } i \in \{1, \ldots, n\} \\ 1 - I_a(g'; C) & \text{if } \alpha(g) = \neg \text{ and } g' \text{ is the direct predecessor of } g \\ \min\big(I_a(g'; C), I_a(g''; C)\big) & \text{if } \alpha(g) = \wedge \text{ and the nodes } g' \text{ and } g'' \text{ are the source nodes of the two incoming edges of } g \\ \max\big(I_a(g'; C), I_a(g''; C)\big) & \text{if } \alpha(g) = \vee \text{ and the nodes } g' \text{ and } g'' \text{ are the source nodes of the two incoming edges of } g \end{cases}$$

Note that the nodes g' and g'' in this definition are not necessarily distinct. Define

$$\begin{aligned} \mathrm{CSAT} = \{C \mid\ & C = (V, E, g_C, \alpha) \text{ is a Boolean circuit with } n \text{ input gates for some } n \in \mathbb{N}^+, \\ & \exists_{a_1, \ldots, a_n \in \{0,1\}}\ I_{a_1, \ldots, a_n}(g_C; C) = 1\}. \end{aligned}$$

CSAT is known to be $\leq_{\mathrm{m}}^{\log}$-complete for NP (proving the membership is straightforward, the hardness can be obtained by a simple reduction from SAT for example). Regarding the encoding, similar conventions as we have introduced for $\mathcal{O}$-circuits result in a precise definition of the length of a Boolean circuit.

Theorem 4.3.9 BC(−) *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP.

Proof The following algorithm yields a reduction from CSAT to BC(−).

1. Input: a Boolean circuit C.
2. We successively transform C into an integer circuit C'. First introduce a new assigned input gate computing $\{1\}$. Then for each inner gate g make the following modifications:
   - If g is a $\neg$-gate with g' as the direct predecessor, then change the label of g to $-$ and add one edge such that $g = \{1\} - g'$.
   - If g is a $\wedge$-gate such that g' and g'' are the (not necessarily distinct) source nodes of the two incoming edges of g, then change the label of g to $-$, delete the two edges from g' and g'' to g, and add two nodes and six edges so that $g = \big(\{1\}-(\{1\}-g')\big)-(\{1\}-g'')$.
   - If g is a $\vee$-gate such that g' and g'' are the (not necessarily distinct) source nodes of the two incoming edges of g, then change the label of g to $-$, delete the two edges from g' and g'' to g, and add two nodes and six edges so that $g = \{1\} - \big((\{1\} - g') - g''\big)$.
3. Return C'.

Observe that the function defined by this algorithm is logarithmic-space computable.

Let us show $C \in \mathrm{CSAT} \Leftrightarrow C' \in \mathrm{BC}(-)$. This equivalence holds if g_C is an input. From now on assume that g_C is not an input gate. Note that both circuits C and C' have the same $n \in \mathbb{N}^+$ unassigned input nodes. Let V denote the set of nodes of C and V' denote the set of nodes of C'. Note $V \subseteq V'$. We obtain inductively that

$$\forall_{\substack{a_1,\ldots,a_n\in\{0,1\},A_1,\ldots,A_n\in\mathcal{P}_{\mathrm{fin}}(\mathbb{N}),\\ \forall_{i\in\{1,\ldots,n\}}\,(a_i=1\Leftrightarrow 1\in A_i)}}\forall_{g\in V}\Big(I_{a_1,\ldots,a_n}(g;C)=1 \Leftrightarrow 1\in I(g;C'(A_1,\ldots,A_n))\Big).$$

Together with the observation that $I(g;C'(A_1,\ldots,A_n)) \subseteq \{1\}$ for each assignment $A_1,\ldots,A_n \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ and each inner gate $g \in V$ this implies $C \in \mathrm{CSAT} \Leftrightarrow C' \in \mathrm{BC}(-)$. □

Theorem 4.3.10 BC(−) ∈ NP.

Proof We sketch an NP-algorithm that accepts BC(−).

1. Input: a $\{-\}$-circuit C with output node g_C and labeling function α.
2. Starting from g_C, go upwards in the circuit always taking the left direct predecessor. Denote the input gate finally reached with g.
3. If g is assigned, then:
   guess an assignment of the unassigned input gates with values from $\mathcal{P}(\alpha(g))$ and accept if the output set is balanced under this assignment, otherwise, reject.
4. Note that now g is unassigned. Let M be the union of all sets computed by assigned input gates and let $m \in \mathbb{N}$ be the least number greater than all numbers in M. Guess an assignment such that $I(g) = \{m\}$ and each unassigned input either computes $\{m\}$ or $\emptyset$. If under this assignment g_C contains m, then accept.
5. Guess an assignment of the unassigned inputs such that each of them computes a set $\subseteq M$. In case g_C is balanced, accept. Otherwise, reject.

Claim 4.3.11 *If the algorithm accepts, then* $C \in \mathrm{BC}(-)$.

Proof If the algorithm accepts in step 3, then clearly $C \in \mathrm{BC}(-)$. If it accepts in step 4, then there is an assignment that maps each unassigned input either to $\{m\}$ or to $\emptyset$ such that m is in the output set. Now change this assignment such that the sets mapped to $\{m\}$ are now mapped to $\{m+1, m+2, \ldots, 2m+1\}$. As no assigned input computes a set that contains a number $\geq m$, it holds $g_C \supseteq \{m+1, m+2, \ldots, 2m+1\}$ under the described assignment. By the choice of g, it can be shown inductively that under every assignment it holds $h \supseteq g$ for each successor h of g, particularly for $h = g_C$. Thus $g_C = g = \{m+1, m+2, \ldots, 2m+1\}$ under the aforementioned assignment and hence $C \in \mathrm{BC}(-)$. Trivially, if C is accepted in step 5, then $C \in \mathrm{BC}(-)$. □

Claim 4.3.12 *If the algorithm rejects, then* $C \notin \mathrm{BC}(-)$.

Proof If the algorithm rejects, then this happens in step 3 or step 5. We argue for the first case. Here g is an assigned input gate. As argued above, under every assignment it holds $g_C \subseteq g$. Hence it suffices to consider assignments that map all unassigned inputs to subsets of $\alpha(g)$. As the algorithm rejects, g_C is not balanced under all these assignments and thus $C \notin \mathrm{BC}(-)$.

It remains to argue for the case where the algorithm rejects in step 5. In this case, g is an unassigned input and as the algorithm does not accept in step 4, there exists no assignment under which the circuit's output set contains a number $\notin M$. Hence it is sufficient to consider assignments that solely map to subsets of M. As the algorithm rejects, none of these assignments yields a balanced output set and hence there exists no assignment at all under which the output set is balanced. Therefore, $C \notin \mathrm{BC}(-)$. □

The observation that the algorithm can be computed in polynomial time by a nondeterministic Turing machine completes the proof. □

Corollary 4.3.13 $\mathrm{BC}(-)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof The assertion follows from Theorem 4.3.9 and Theorem 4.3.10. □

Theorem 4.3.14 $\mathrm{BC}(\emptyset) \in \mathrm{L}$.

Proof Let C be an $\emptyset$-circuit. Then the output node is an input node. If the output node is unassigned, then $C \in \mathrm{BC}(\emptyset)$. Otherwise, $C \in \mathrm{BC}(\emptyset)$ if and only if the set computed by the output node is balanced. Thus by Proposition 4.1.2, the proof is complete. □

4.4 Summary and Discussion

The following table summarizes our results, namely the lower and upper bounds for the complexity of $\mathrm{BC}(\mathcal{O})$ with $\mathcal{O} \subseteq \{-, \cdot\}$. As each problem $\notin \{\emptyset, \overline{\emptyset}\}$ trivially is $\leq_{\mathrm{m}}^{\log}$-hard for L, it is not necessary to prove the mentioned lower bound for $\mathrm{BC}(\emptyset)$.

$\mathrm{BC}(\mathcal{O})$ for $\mathcal{O} =$	$\leq_{\mathrm{m}}^{\log}$-hard for	contained in
$\emptyset$	L	L, Theorem 4.3.14
$\{-\}$	NP, Theorem 4.3.9	NP, Theorem 4.3.10
$\{\cdot\}$	NL, Theorem 4.3.7	NL, Theorem 4.3.4
$\{\cdot, -\}$	undecidable, Theorem 4.2.1	

To our knowledge, in contrast to all results from previous papers on complexity issues concerning decision problems for integer circuits (e.g., [MW07, Tra06, Bre07, GHR+10, GRTW10, BBD+20]) or related constraint satisfaction problems ([GJM17, Dos16]), a problem *admitting only one arithmetic operation* is shown to be undecidable. Beginning with this problem, namely $\mathrm{BC}(-,\cdot)$, the problems $\mathrm{BC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{-,\cdot\}$ are systematically investigated and for each of these problems the computational complexity is precisely characterized. It turns out that decreasing the size of the set of allowed operations yields problems that are in NP. In particular, all these problems are $\leq_{\mathrm{m}}^{\log}$-complete for one of the classes L, NL, and NP.

As a side effect, we obtain as a corollary of our main result, the undecidability of $\mathrm{BC}(-,\cdot)$, that the problems $\mathrm{BC}(\triangle,\cup,\cdot)$ and $\mathrm{BC}(\triangle,\cup,\cdot)$ are undecidable as well.

As the complexity of the four problems is precisely characterized, this chapter is a complete unit in some sense. Nevertheless, there arise new questions from our results:

1. Is there a set $\mathcal{O} \subseteq \{-,\cup,\cap\}$ such that $\mathrm{BC}(\mathcal{O} \cup \{+\})$ is undecidable?
2. $\mathrm{BC}(\triangle,\cdot) \in \mathrm{REC}$?

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160:781–793, 2004.

[BBD+17] D. Barth, M. Beck, T. Dose, C. Glaßer, L. Michler, and M. Technau. Emptiness problems for integer circuits. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

[BBD+20] D. Barth, M. Beck, T. Dose, C. Glaßer, L. Michler, and M. Technau. Emptiness problems for integer circuits. *Theor. Comput. Sci.*, 824–825:11–35, 2020.

[Bey04] O. Beyersdorff. Representable disjoint NP-pairs. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2004)*, volume 3328 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2004.

[Bey06] O. Beyersdorff. Disjoint NP-pairs from propositional proof systems. In *Proceedings of the Third International Conference on Theory and Applications of Models of Computation*, volume 3959 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2006.

[Bey07] O. Beyersdorff. Classes of representable disjoint NP-pairs. *Theoretical Computer Science*, 377(1-3):93–109, 2007.

[Bey10] O. Beyersdorff. The deduction theorem for strong propositional proof systems. *Theory of Computing Systems*, 47(1):162–178, 2010.

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

[BKM09] O. Beyersdorff, J. Köbler, and J. Messner. Nondeterministic functions and the existence of optimal proof systems. *Theor. Comput. Sci.*, 410(38-40):3839–3855, 2009.

[Bre07] H. Breunig. The complexity of membership problems for circuits over sets of positive numbers. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT 2007)*, volume 4639 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2007.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158. ACM, 1971.

[CR79] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.

[DG19] T. Dose and C. Glaßer. NP-completeness, proof systems, and disjoint NP-pairs. *Electron. Colloquium Comput. Complex.*, 26:50, 2019.

[DG20] T. Dose and C. Glaßer. NP-completeness, proof systems, and disjoint NP-pairs. In *Proceedings of the 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

[Dos16] T. Dose. Complexity of constraint satisfaction problems over finite subsets of natural numbers. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics*, pages 32:1–32:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

[Dos18] T. Dose. Balance problems for integer circuits. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

[Dos19a] T. Dose. Balance problems for integer circuits. *Theor. Comput. Sci.*, 799:124–139, 2019.

[Dos19b] T. Dose. P-optimal proof systems for each non-empty NP-set but no complete disjoint NP-pairs relative to an oracle. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

[Dos20a] T. Dose. an oracle separating conjectures about incompleteness in the finite domain. *Theor. Comput. Sci.*, 809:466–481, 2020.

[Dos20b] T. Dose. further oracles separating conjectures about incompleteness in the finite domain. *Theor. Comput. Sci.*, 847:76–94, 2020.

[DPR61] M. Davis, H. Putnam, and J. Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics*, 74(2):425–436, 1961.

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

[EY80] S. Even and Y. Yacobi. Cryptocomplexity and NP-completeness. In *Proceedings of the 7th International Colloquium on Automata, Languages and Programming (ICALP 1980)*, volume 85 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 1980.

[For08a] K. Ford. Integers with a divisor in $(y, 2y]$. In *Anatomy of integers*, volume 46 of *CRM Proc. and Lect. Notes*, pages 65–81. Amer. Math. Soc., Providence, RI, 2008.

[For08b] K. Ford. The distribution of integers with a divisor in a given interval. *Annals of Math. (2)*, 168:367–433, 2008.

[GHR+10] C. Glaßer, K. Herr, C. Reitwießner, S. D. Travers, and M. Waldherr. Equivalence problems for circuits over sets of natural numbers. *Theory of Computing Systems*, 46(1):80–103, 2010.

[GJM17] C. Glaßer, P. Jonsson, and B. Martin. Circuit satisfiability and constraint satisfaction around Skolem Arithmetic. *Theor. Comput. Sci.*, 703:18–36, 2017.

[GNW90] T. Gundermann, N. A. Nasser, and G. Wechsung. A survey on counting classes. In *Proceedings of the Fifth Annual Structure in Complexity Theory Conference*, pages 140–153. IEEE Computer Society, 1990.

[GRTW10] C. Glaßer, C. Reitwießner, S. D. Travers, and M. Waldherr. Satisfiability of algebraic circuits over sets of natural numbers. *Discrete Applied Mathematics*, 158(13):1394–1403, 2010.

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

[GSS05] C. Glaßer, A. L. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. *Information and Computation*, 200:247–267, 2005.

[GSSZ04] C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.

[GSZ07] C. Glaßer, A. L. Selman, and L. Zhang. Canonical disjoint NP-pairs of propositional proof systems. *Theoretical Computer Science*, 370:60–73, 2007.

[GSZ09] 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.

[HH88] J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theor. Comput. Sci.*, 58:129–142, 1988.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Proceedings of a Symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[Kha19] E. Khaniki. New relations and separations of conjectures about incompleteness in the finite domain. *arXiv e-prints*, page arXiv:1904.01362, Apr 2019.

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

[KMT03] J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184(1):71–92, 2003.

[Kou14] D. Koukoulopoulos. On the number of integers in a generalized multiplication table. *Journal für die reine und angewandte Mathematik*, 689:33–99, 2014.

[KP89] 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.

[Lov79] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

[Mat70] Y. V. Matiyasevich. Enumerable sets are diophantine. *Doklady Akad. Nauk SSSR*, 191:279–282, 1970. Translation in Soviet Math. Doklady, 11:354–357, 1970.

[Min67] M. L. Minsky. *Computation: Finite and Infinite Machines*. Automatic Computation. Prentice–Hall, 1967.

[MP91] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991.

[MW07] P. McKenzie and K. W. Wagner. The complexity of membership problems for circuits over sets of natural numbers. *Computational Complexity*, 16(3):211–244, 2007.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory of feasible closure properties. *Journal of Computer and System Sciences*, 46:295–325, 1993.

[Pap94] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[PD09] I. Pratt-Hartmann and I. Düntsch. Functions definable by arithmetic circuits. In *Proceedings of the 5th Conference on Computability in Europe (CiE 2009)*, volume 5635 of *Lecture Notes in Computer Science*, pages 409–418. Springer, 2009.

[Pud96] P. Pudlák. On the lengths of proofs of consistency. In *Collegium Logicum*, pages 65–86. Springer Vienna, 1996.

[Pud03] P. Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theoretical Computer Science*, 295:323–339, 2003.

[Pud13] P. Pudlák. *Logical Foundations of Mathematics and Computational Complexity - A Gentle Introduction*. Springer monographs in mathematics. Springer, 2013.

[Pud17] P. Pudlák. Incompleteness in the finite domain. *The Bulletin of Symbolic Logic*, 23(4):405–441, 2017.

[Rac82] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the ACM*, 29:261–268, 1982.

[Raz94] A. A. Razborov. On provably disjoint NP-pairs. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(6), 1994.

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

[Sip19] M. F. Sipser. Adventures in complexity. http://www.fields.utoronto.ca/video-archive/2019/05/2774-20374, 2019. Accessed: 2020-06-30.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC 1973)*, pages 1–9. ACM, 1973.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inf.*, 26(3):279–284, 1988.

[Tar88] E. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[Tra06] S. D. Travers. The complexity of membership problems for circuits over sets of integers. *Theoretical Computer Science*, 369(1-3):211–229, 2006.

[Tur37] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1937.

[Val76] L. G. Valiant. Relative complexity of checking and evaluation. *Information Processing Letters*, 5:20–23, 1976.

[Ver91] O. V. Verbitskii. Optimal algorithms for coNP-sets and the EXP =? NEXP problem. *Mathematical notes of the Academy of Sciences of the USSR*, 50(2):796–801, Aug 1991.

[Wag84] K. W. Wagner. The complexity of problems concerning graphs with regularities (extended abstract). In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS 1984)*, pages 544–552, 1984.

[Yan01] K. Yang. Integer circuit evaluation is pspace-complete. *J. Comput. Syst. Sci.*, 63(2):288–303, 2001.

Index