The Power of Counting and Exponential Sums

Dissertation zur Erlangung des Grades des Doktors der Naturwissenschaften (Dr. rer. nat.) der Fakultät für Mathematik und Informatik der Universität des Saarlandes

vorgelegt von

Julian Dörfler

Saarbrücken,
2026

Kolloquium Informationen

Datum	16. Juni 2026; 17:00 CEST
Dekan	Prof. Dr. Jan Reineke
Vorsitzender	Prof. Dr. Jan Reineke
Gutachter	Prof. Dr. Markus Bläser
	Prof. Dr. Joël Ouaknine
	Prof. Dr. Thomas Icard
	Prof. Dr. Greta Panova
Akademischer Mitarbeiter	Dr. Anurag Pandey

Acknowledgments

Firstly, I want to thank Markus Bläser for funding and supervising me all these years. Without his support and introductions to many facets of complexity theory there simply is no way I'd be where I am today.

Secondly, I want to thank Christian Ikenmeyer for first introducing me to geometric complexity theory – even if it ended up not being part of this thesis – and later to the concept of combinatorial interpretations. Moreover I want to thank him for all the fruitful collaborations.

Thirdly, I want to thank Holger Dell and Marc Roth for introducing me to graph motif parameters and the parametrized complexity world. Moreover I want to thank both of them for the fruitful collaborations.

Fourthly, I want to thank Maciej Liśkiewicz for introducing me to the concepts of causality and all the fruitful collaborations on it.

Fifthly, I want to thank the other co-authors of my publications, Radu Curticapean, Gorav Jindal, Greta Panova, Johannes Schmitt, Philip Wellnitz and Benito van der Zander, for the collaborations and all the tricks and ideas I was able to learn from them.

Sixthly, I want to thank Miriam Backens, Marian Dietz, Jacob Focke, Anurag Pandey and Marcel Wienöbst for collaborations with me over the years on projects that have not (yet) seen the light of day.

Seventhly, I want to thank Thomas Icard, Joël Ouaknine and Greta Panova for taking the time to review this thesis.

Eightly, I want to give a special thank you to Wolfgang Pohl, the BWINF, all the IOI training coaches and my fellow competitors and friends in these contests for igniting my interest in algorithms and theory.

Ninthly, I want to thank my friends Alex, Dominic, Kai, Marcel and Simon[2] for their support and the interesting and insightful discussions over the years.

Lastly, I want to thank my parents for allowing and encouraging me to pursue my interests in computer science freely and from an early age and supporting me throughout my studies.

Summary

We first study the concept of "Combinatorial Interpretations", i.e. the question of which quantities "count something". We start by showing that counting using finite automata is precisely closed under functions that ultimately are polynomial on residue classes and sums/products thereof. Following this, we show that graph motif parameters $f(G) = \sum_i \alpha_i \#\mathrm{Ind}(H_i \to G)$, where all pattern graphs H_i are without isolated vertices, can be counted by nondeterministic Turing machines in a local way exactly when all coefficients α_i are non-negative integers. The result is then generalized to other forms of subobject counts such as more general relational structures, colored graphs, finite vector spaces over finite fields, and parameter sets using category theory.

Secondly, we study the complexity theoretic impact of adding unary summation primitives to the existential theory of the reals. We observe an exponential jump in complexity when the summation variables are allowed to be used to index variables, thus allowing for an exponential number of variables. This jump is characterized by the machine model of nondeterministic real word RAMs using exponential time, versus the polynomial time required for the standard existential theory of the reals. However, if the summation variables cannot be used for variable indexing, then we only observe $\mathsf{NP}_{\mathrm{real}}^{\mathsf{VNP}_{\mathbb{R}}}$-completeness. It is thus likely a weaker model than the additional models compressing the input formula that we investigate: both an additional unary product primitive and a succinct encoding allowing only for polynomially many variables each lead to PSPACE-completeness.

Next, we add unary summation primitives to the logic for reasoning about probabilities introduced by Fagin et al. [FHM90] – extended by Ibeling and Icard [II20] to reason about Pearl's causal hierarchy. We fully investigate the changing landscape of complexity classes, depending on the arithmetic power of the terms, the level of the causal hierarchy, and varying constraints on the model. Many of these variations form natural complete problems for the complexity classes we observe by augmenting the existential theory of the reals.

Lastly, we study the complexity of the identification problem on linear structural causal models. We are both interested in the complexity of numerically determining the parameters of the model uniquely, given the covariance matrix of the random variables, and the complexity of the generic problem of determining, for a given causal diagram, whether it is possible to reconstruct the parameters uniquely in almost all cases.

Zusammenfassung

Wir betrachten zuerst das Konzept der "kombinatorischen Interpretierbarkeit", d.h. die Frage, welche Größen "etwas zählen". Wir beginnen damit zu zeigen, dass Zählen mithilfe von endlichen Automaten abgeschlossen ist unter genau solchen Funktionen, die letztendlich PORC-Funktionen sind und Summen/Produkte davon. Anschließend zeigen wir, dass Graph Motif Parameter $f(G) = \sum_i \alpha_i \#\mathrm{Ind}(H_i \to G)$, bei denen alle Mustergraphen H_i frei von isolierten Knoten sind, von einer nichtdeterministischen Turingmaschine genau dann lokal gezählt werden können, wenn alle Koeffizienten α_i nicht-negative Ganzzahlen sind. Dieses Ergebnis verallgemeinert sich zu anderen Arten von Subobjekten wie allgemeinen relationalen Strukturen, gefärbten Graphen, endlichen Vektorräumen über endlichen Körpern und Parametermengen durch die Verwendung von Kategorientheorie.

Des Weiteren studieren wir den komplexitätstheoretischen Einfluss eines unären Summensymbols, zuerst zur existenziellen Theorie der reellen Zahlen. Wir beobachten einen exponentiellen Sprung in der Komplexität, wenn die Summationsvariablen verwendet werden dürfen, um Variablen zu indizieren, was eine Verwendung von exponentiell vielen Variablen erlaubt. Dieser Sprung lässt sich charakterisieren durch das Maschinenmodell der nichtdeterministischen reellen Wort-RAM mit exponentieller Zeit im Vergleich zur polynomiellen Zeit, die für die normale existenzielle Theorie der reellen Zahlen nötig ist. Falls jedoch die Summationsvariablen nicht zum Indizieren von Variablen verwendet werden dürfen, dann erhalten wir $\mathsf{NP}_{\mathrm{real}}^{\mathsf{VNP}_{\mathbb{R}}}$-Vollständigkeit. Es handelt sich daher wahrscheinlich um ein schwächeres Modell als die anderen Kompressionsarten, die wir betrachten: ein unäres Produktsymbol, sowie kompakte Kodierungen, bei denen nur polynomiell viele Variablen verwendet werden dürfen, führen beide zu PSPACE-Vollständigkeit.

Als nächstes fügen wir das Summensymbol in die Logik zum Schlussfolgern über Wahrscheinlichkeiten von Fagin et al. [FHM90] – erweitert von Ibeling and Icard [II20] für Pearls kausale Hierarchie – ein. Wir analysieren die Landschaft der Komplexitätsklassen, die sich durch verschieden starke Arithmetik, die kausale Ebene und verschiedener Beschränkungen des Modells ergibt, vollständig. Viele dieser Variationen formen natürliche vollständige Probleme für die Komplexitätsklassen, die wir auch durch unsere Veränderungen der existenziellen Theorie der reellen Zahlen beobachten.

Zuletzt betrachten wir die Komplexität des Identifikationsproblems auf linearen strukturellen Kausalmodellen. Wir interessieren uns sowohl für die Komplexität, um, gegeben die Kovarianzmatrix der Zufallsvariablen, numerisch die Parameter des Modells eindeutig zu bestimmen, als auch die Komplexität, um zu entscheiden, gegeben ein kausales Diagramm, ob sich die Parameter in fast allen Fällen eindeutig rekonstruieren lassen.

Outline of this Thesis

Publications and Collaborators

This thesis is based on the publications [DI24; BDLZ24; DZBL24; DZBL25; BDLZ25; BCDI25]. Moreover, Chapters 1 to 3, 7 and 10 are exclusively meant to aid the understanding of the reader, while Chapters 4 to 6, 8 and 9 are to be considered the main contributions of this thesis. The precise publications each chapter are based on are as follows:

Chapter 1 An informal introduction to the topics of this thesis.

Chapter 2 A grouping of shared preliminaries for this thesis, in particular containing the relevant complexity classes. Some of these preliminaries are taken from various of the papers listed above.

Chapter 3 An introduction to the ideas of "Combinatorial Interpretations". Contains some paragraphs from [DI24; BCDI25].

Chapter 4 An analysis of the functional closure properties of #FA, closely based on [DI24]. This work is joint work with Christian Ikenmeyer: the main contributions are those of the author, while Christian Ikenmeyer held a supervisional role, contributing primarily to the background on combinatorial interpretations and some presentation.

Chapter 5 This chapter introduces a dichotomy about which graph motif parameters can be considered as local combinatorial interpretations, closely based on [BCDI25]. This work is joint work with Markus Bläser, Radu Curticapean and Christian Ikenmeyer: the main contributions are those of the author, while Markus Bläser, Radu Curticapean and Christian Ikenmeyer held a supervisional role, contributing primarily to presentation and background on Type-2 complexity, graph motif parameters and combinatorial interpretations respectively.

Chapter 6 This chapter investigates variations of the existential theory of the reals with the addition of summation operators and other succinct representations. It is based on [BDLZ24]. This work is joint work with Markus Bläser, Maciej Liśkiewicz and Benito van der Zander: the main contributions that are part of this chapter are those of the author, while Benito van der Zander did a lot of editing and reviewing and Markus Bläser and Maciej Liśkiewicz held a supervisional role, contributing primarily to presentation and background on topics regarding algebraic complexity and causality respectively.

Chapter 7 This chapter introduces the well known concepts of structural causal models and Pearl's causal hierarchy. Some of the text and the example are based on [BDLZ25] and [DZBL25].

Chapter 8 This chapter investigates the complexity of satisfiability problems over structural causal models with many different parameters. It is based on results from [BDLZ24], [DZBL25] and [BDLZ25]. This chapter majorly restructures and unifies these results and extends them in multiple ways: Firstly, it completes the picture of the impact of small models as a constraint on the model. Secondly, it adds multiple different interpretations of causal diagrams, the semi-Markovian and full interpretations, while completing the picture for the Markovian interpretation. The original works are joint work with Markus Bläser, Maciej Liśkiewicz and Benito van der Zander: the main contributions are equal contributions between the author and Benito van der Zander, while Markus Bläser and Maciej Liśkiewicz held a supervisional role, contributing primarily to presentation and background on topics regarding algebraic complexity and causality respectively.

Chapter 9 This chapter analyzes the complexity of the identification problem for linear structural causal models. It is based on [DZBL24]. The original work is joint work with Markus Bläser, Maciej Liśkiewicz and Benito van der Zander: the main contributions are equal contributions between the author and Benito van der Zander, while Markus Bläser and Maciej Liśkiewicz held a supervisional role, contributing primarily to presentation and background on topics regarding algebraic complexity and causality respectively. However, this chapter primarily focuses on the results that are a contribution of the author, i.e. numeric and generic identifiability. The results on cyclic graphs and edge identifiability are not part of this thesis.

Chapter 10 This chapter gives a very short overview over the results shown in this thesis and some potential directions for further research.

Recommended Reading Order

While I would of course recommend you, the reader, to read this entire thesis[1] front to back, I understand if you prefer to only look at some chapters. To aid you in your endeavors and avoid missing some prerequisites, I am providing you with the rough dependencies of the chapters on each other. Note that some of these are not hard dependencies and depending on your level of prior knowledge about these topics might be redundant. For convenience, this dependency chart will reappear at the start of each chapter, with the relevant dependencies highlighted.

[1] After all, all topics in this thesis are interesting! But I might be a bit biased as the author.

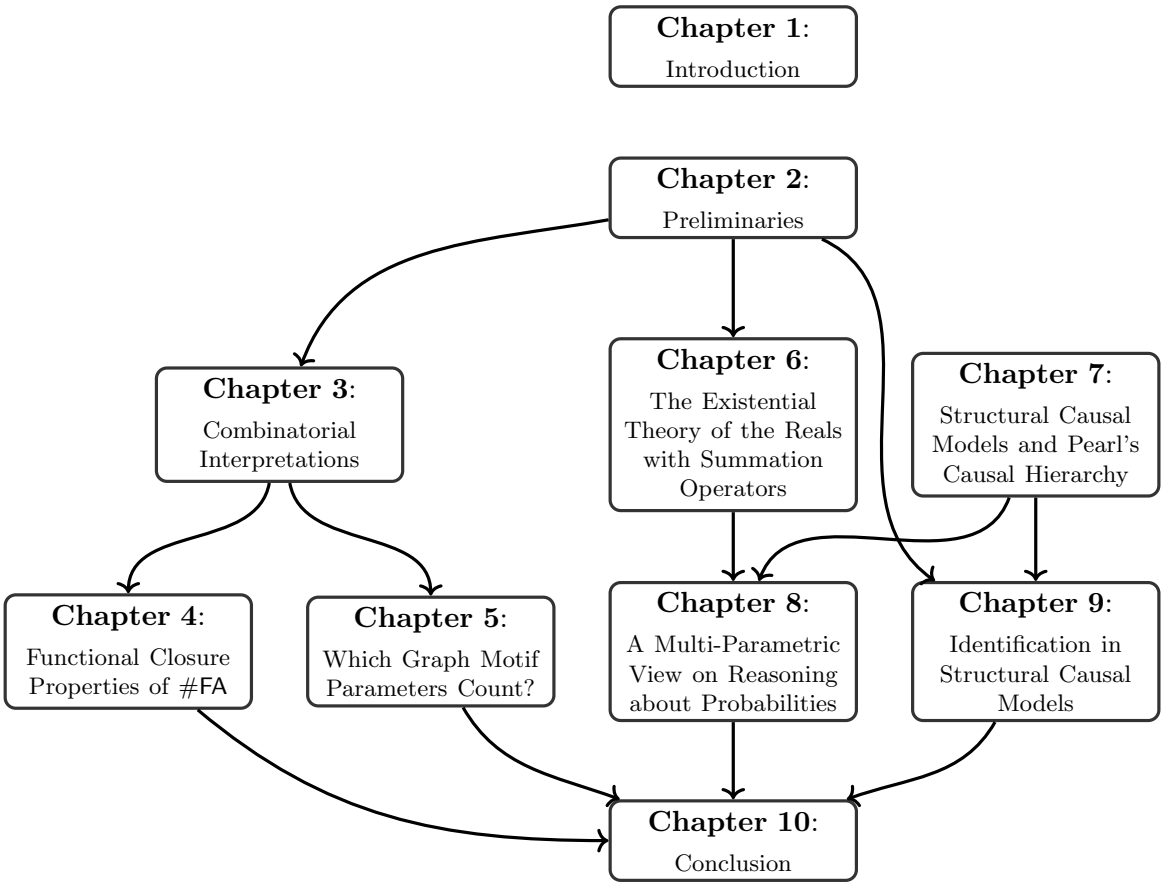


Figure 1: Dependencies within this thesis. Transitive dependencies are omitted.

UNIVERSITÄT DES SAARLANDES

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Declaration of original authorship

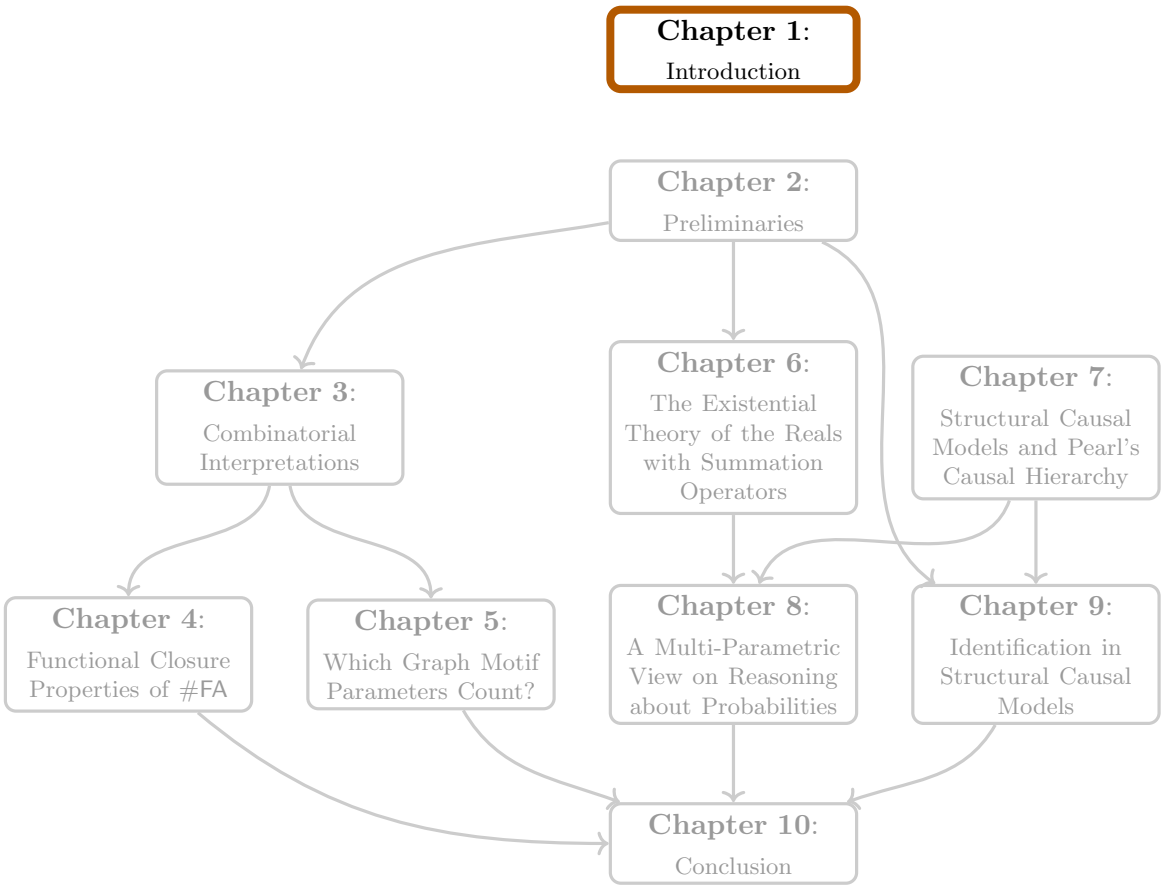
I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Saarbrücken, February 2026

gez. / signed

Julian Dörfler

Contents



Introduction

1.1 What is Counting?

In simplest terms, counting is the process of determining the quantity of finitely many discrete objects. We can count sheep, we can count people on earth[1], however we cannot count the water on earth[2], but what about some more abstract quantities?

Given a finite set S with n elements, can we count the number of elements in S? Of course we can, it's just n. While this question seems obvious, how about the number of singleton subsets of S? This is just the same question rephrased, so the answer clearly is still n. Once we generalize to subsets of size k, we get that there are $\binom{n}{k}$ subsets of size k in S. What does this tell us about $\binom{n}{k}$? It tells us $\binom{n}{k} = |\{S' \subseteq S \mid |S'| = k\}|$ denotes the size of a "natural" set, so in particular it *counts* something. In a similar way, if we have another set T with m elements, then $n + m$ counts the number of elements in the disjoint union $S \uplus T$, $n \cdot m$ counts the number of elements in the cartesian product $S \times T$, and 2^n counts the number of subsets of S of arbitrary size. Inductively repeating these constructions, we see that $\binom{n \cdot m}{k}$ also counts the size of a set, namely, the size of $\{A \subseteq S \times T \mid |A| = k\}$.

While this view is great it also has its limits: It allows us to consider any non-negative integer quantity as counting! To justify some expression x to count something we could try and argue that it is simply the size of the set $\{n \in \mathbb{N} \mid n < x\}$. We need some way of restricting what kinds of conditions we consider as "natural" or "simple". The currently best attempt for formalizing this, pioneered by Igor Pak, argues that the answer should be any condition that can be verified in polynomial time by a Turing machine, i.e. the complexity class #P. See his survey [Pak23] for his arguments of why this captures combinatorial interpretability.

But let us back up first, how did we arrive at a complexity class of functions instead of just a single quantity? Our considered quantities are not just a single number, but instead an indexed family of numbers. In our examples above, they are indexed by the natural numbers n and m, however we also allow more complicated indexing, for example indexed by graphs. We then want to compute $f(x) = |A_x|$ for some set $A_x = \{y \mid P(x, y)\}$ with a predicate/condition P. If we quantify $x, y \in \{0,1\}^\star$ and require P to be computable in time polynomial in x, then f is combinatorial iff $f \in \#\mathsf{P}$. Thus, equivalently, we can also consider A_x to be the set of accepting paths of a fixed nondeterministic Turing machine running in polynomial time in $|x|$. While this definition of combinatorial interpretations seems natural and is the most studied one, it is not without flaws: On one hand, it is too broad: Any counting function $f \in \mathsf{FP}$, i.e. a function computable in polynomial time, is also in #P. However, the resulting interpretation $f(x) = |A_x| = |\{y \in \mathbb{N} \mid y < f(x)\}|$ arguably does not tell us how to interpret f in any combinatorial way, even though everything is computable in polynomial time. On the other hand, it is too restrictive: The power set $\mathcal{P}(S) = \{S' \subseteq S\}$, corresponding to the function $n \mapsto 2^n$, is an important primitive in combinatorics. However $n \mapsto 2^n$ is not computable in #P, if n is given in binary. Even if n is given in unary, $n \mapsto 2^{2^n}$, corresponding to $\mathcal{P}(\mathcal{P}(S))$, is not computable in #P.

[1] Of course there are problems like fluctuations in the amount of people that would occur during your counting, but if you were fast enough, in theory, you could count the exact amount of humans alive at any given moment.

[2] We consider water to be non-discrete here, but if you really wanted to count something you could count the number of water molecules instead.

In Chapter 3 we go into a bit more detail about combinatorial interpretations and introduce the term *local combinatorial interpretations*. In a local combinatorial interpretation the predicate above is forbidden from observing all of x and instead is only allowed to see a poly-logarithmic fragment of it. We then use this definition in Chapter 5 to investigate which "subobject counts" have a local combinatorial interpretation.

A second important primitive in the context of combinatorial interpretations are functional closure properties of complexity classes $\mathcal{C}$, i.e. the functions $\varphi : \mathbb{N}^m \to \mathbb{N}$ with $\varphi(\mathcal{C}^m) \subseteq \mathcal{C}$. They often directly correspond to combinatorial proofs (see Section 3.3 for the connection). In Chapter 4 we fully characterize the functional closure properties of $\#\mathsf{FA}$, the class of all counting functions f such that a nondeterministic finite automaton M has exactly $f(w)$ accepting paths on input w.

1.2 Exponential Sums

The second main topic of this thesis is exponential sums. Consider the term $T = x_{0,0} + x_{0,1} + x_{1,0} + x_{1,1}$. We can write T more compactly as $\sum_{a=0}^{1} \sum_{b=0}^{1} x_{a,b}$. Allowing for this unary sum operator allows us to write some terms that would require exponential size much more compactly, inspiring the name *exponential sums*.

They form a natural generalization of counting: If we have some predicate $P(x, y)$ and would like to know the number of $y \in \{0,1\}^{p(n)}$ s.t. $P(x, y) = 1$, this is equivalent to calculating $\sum_{y_1=0}^{1} \cdots \sum_{y_{p(n)}=0}^{1} P(x, y_1 \cdots y_{p(n)})$, if we assume $P(x, y) \in \{0, 1\}$.

Counting solutions is generally assumed to be much harder than just checking for the existence of a single solution. As such Toda's theorem [Tod91] informally states that if you have an oracle that can give you the number of solutions to some polynomial time predicate, you can efficiently decide the entire polynomial time hierarchy. The polynomial time hierarchy is believed to strictly contain checking for the existence of single solutions.

Thus, a natural question is, how do exponential sums influence the complexity of problems that take as input some terms? The biggest increase in complexity possible is an exponential jump: An algorithm aiming to solve the problem with exponential sums can always remove the exponential sums by just expanding the input explicitly and then using the algorithm designed for the problem without exponential sums. On the other hand, it is possible that exponential sums do not increase the complexity. One such example we will see has terms that are allowed to contain *exponential products*. Additionally adding in exponential sums, however, does not increase the complexity further (see Section 6.5).

It turns out that, probably not too surprisingly, the precise change in complexity heavily depends on the specific problem and the allowed terms, but in most cases we investigate the complexity does indeed increase.

The first type of terms we are interested in are Boolean combinations of polynomial (in-)equalities in some real variables $x_1, \ldots, x_n$. The corresponding problem is then the question whether there is any assignment of real values to $x_1, \ldots, x_n$, s.t. the resulting sentence is true, known as the existential theory of the reals (ETR). We introduce the theory of the reals in more detail in Section 2.5.5.

In Chapter 6 we then see how different ways of compactly representing ETR formulas influence the complexity and investigate the complexity classes resulting from this. Among others, we look at

- using exponential sums where summation variables are allowed to index variables (one such example of variable indexing is the formula at the start of this section),
- using exponential sums where summation variables are not allowed to index variables,
- using exponential products, and
- compressing the formula using Boolean circuits.

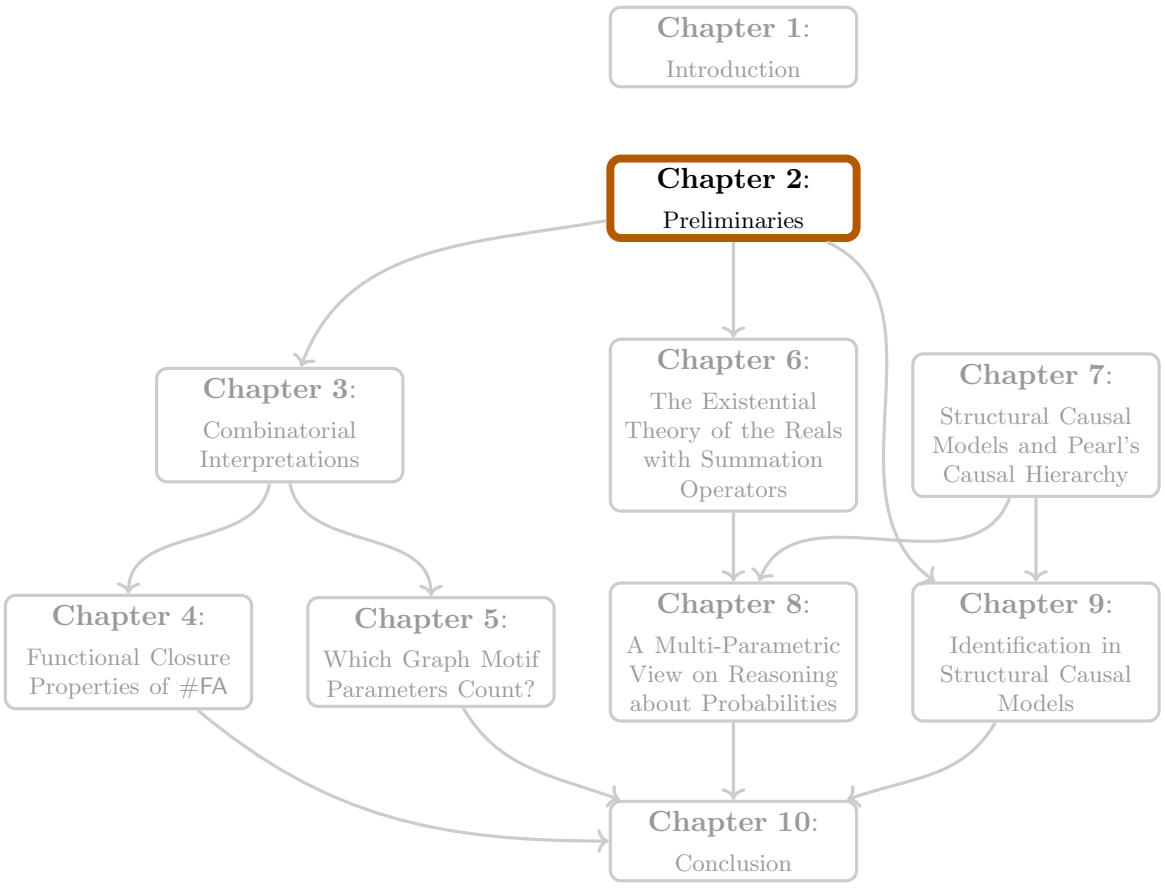
The second type of terms we investigate are Boolean combinations of arithmetic combinations of discrete probabilistic terms $\mathbb{P}(X{=}3)$, $\mathbb{P}(X{=}2 \vee Y{=}1)$, $\mathbb{P}(X{=}2 \wedge Y{=}1)$, etc. Here, exponential sums allow marginalizing variables. For example, $\mathbb{P}(X{=}3) = \sum_y \mathbb{P}(X{=}3 \wedge Y{=}y)$ allows to calculate the probability $\mathbb{P}(X{=}3)$ when only access to the joint probability distribution of X and Y is given. The corresponding problems are then generally of the form "Is there a probability distribution such that the resulting sentence is true?"

While basic probabilities are great at describing correlations between different events, they are unable to explain why. Using them can tell us that "The grass is wet" and "It is raining" will certainly be correlated, however it can never tell us which of these influences the other. Human intuition tells us that it must be the rain making the grass wet instead of wet grass causing rain, but not in every case the causal relationships are clear. One common example is the following situation: If I ask you, what is the causal relationship between "There are more ice cream sales" and "There is more theft", you would most likely say "None of these influences the other" and you would be correct. Yet there is a correlation between the two. The causal relationships only become clear once we introduce a third random variable "The weather is good": Now, good weather increases ice cream sales and it increases theft by virtue of more people being outside.

Pearl's causal hierarchy [Pea09] introduces ways to formally talk about such causal relationships and how to model processes that have causal information via structural causal models. We introduce both of these in Chapter 7.

In Chapter 8 we then investigate how the different levels of Pearl's causal hierarchy, combined with different arithmetic operations (sometimes including and sometimes excluding exponential sums) and further restrictions on the models influences the complexity of the question "Is there a probability distribution that fulfills our input formula?"

Finally, in Chapter 9 we look at the question "Given the causal structure of a process and observational data, can we uniquely recover the exact strengths of the causal influences?"



Preliminaries

In this chapter we give a brief overview over the common notations and complexity classes we will need throughout this work. To get a more detailed overview over the computational models and complexity classes we recommend [HO02].

2.1 Terminology

Throughout we have $\mathbb{N} = \{0, 1, 2, \ldots\}$ and use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

Fix some finite *alphabet* Σ. The elements of Σ are called *symbols*. A *word* (or *string*) $w = (w_1, \ldots, w_\ell)$ is then a finite sequence of symbols from Σ. Usually it is abbreviated as $w = w_1 \ldots w_\ell$, so for example 0000, 101010 and 1 are words over the binary alphabet $\Sigma = \{0, 1\}$. The length of w is denoted as $|w| = \ell$ and the set of all words of that size are denoted Σ^ℓ. The unique word of length zero is denoted ε. The set of all words is denoted $\Sigma^\star = \bigcup_{i \in \mathbb{N}} \Sigma^i$. The set of all words with length at most j is denoted $\Sigma^{\leq j} = \bigcup_{i=0}^{j} \Sigma^i$.

A *language* $L \subseteq \Sigma^\star$ is then some subset of words, while *counting functions* $f : \Sigma^\star \to \mathbb{N}$ assign each individual word a nonnegative integer. The complement $\overline{L}$ of a language L is the set of all words that are not part of L, i.e. $\overline{L} = \Sigma^\star \setminus L$. A complexity class then usually consists of either a set of languages or a set of counting functions. We sometimes use the umbrella term *problem* for languages, counting functions and functions in general we'd like to know whether they can be computed by certain models.

2.2 Models of Computation

Most of our complexity classes are defined using a couple of different machine models: finite automata, Turing machines (TMs), real word random access machines (real word RAMs) and circuits/formulas. We give a brief introduction and explanation of each of them.

2.2.1 Finite Automata

We start with the simplest of our models. Imagine you want to determine whether a number given in binary[1] is even, by only iterating through the input word exactly once from left to right. Further you only have finite storage space available. In the end your answer may only depend on what is stored in said storage space.

[1] For simplicity we allow arbitrary amounts of leading zeros, including allowing ε as a representation for the number 0.

There is of course a simple algorithm that fits this description, based on the fact that a binary number is even iff the last digit is 0: Initially the storage contains 0. Then iterate through the input word and always store the last symbol you have seen. At the end, you return "YES" if your stored digit is a 0 and "NO" otherwise.

What do we need to specify to define such an algorithm?

- The possible contents of our finite storage space.
- The possible symbols appearing in the input.
- A procedure that, given a symbol and the content of the storage space, returns the new content of the storage space.
- The initial content of the storage space.
- A translation from the content of the storage space to acceptance.

From now on we call the finite storage space the finite state space. This gives rise to the following definition of a deterministic finite automaton:

2.1 Definition. A *deterministic finite automaton (DFA)* M is a tuple $(Q, \Sigma, \delta, q_0, Q_{\text{acc}})$ consisting of

- a finite set of states Q,
- a finite alphabet Σ,
- a partial transition function $\delta : Q \times \Sigma \to Q$,
- an initial state $q_0 \in Q$ and
- a set of accepting states $Q_{\text{acc}} \subseteq Q$.

A *computation* (sometimes also called *computation path* or just *path*) of M on some word $w \in \Sigma^\star$ is then a sequence of states $q_0, \ldots, q_{|w|} \in Q$, such that q_0 is the initial state and $q_i = \delta(q_{i-1}, w_i)$ for all $i \in \{1, \ldots, |w|\}$. We call such a computation accepting if $q_{|w|} \in Q_{\text{acc}}$ and rejecting otherwise.

Further we say M accepts w if the computation of M on w is accepting, otherwise M rejects w. Remember that δ can be partial, so it is possible there is no computation of M on w, in that case we also say that M rejects w. This way M recognizes a language

$$L(M) := \{w \in \Sigma^\star \mid M \text{ accepts } w\}.$$

To simplify the presentation of finite automata we often use a graphical approach. The different states are drawn as circles, the initial state is marked with an incoming arrow and accepting states are circled twice. Transitions are drawn as arrows between the states, each marked with the corresponding symbol that was read during the transition, potentially comma separated if there are multiple transitions between the same two states, i.e. states q, q' have an arrow $q \xrightarrow{\sigma} q'$ if $\delta(q, \sigma) = q'$. The algorithm from the start of the section would then be visualized as in Figure 2.1.

In a deterministic finite automaton, all transitions are deterministic, i.e. given a symbol to be read and a state the DFA is in, there is a unique successor state (if any). This, however, is a problem when we want to use finite automata for counting. A natural way for counting (as is the case for #P, see Section 2.5.2) is to count the number of accepting

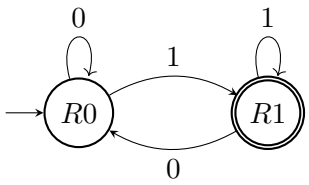


Figure 2.1: A DFA accepting precisely the even binary numbers.

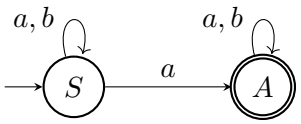


Figure 2.2: NFA counting the number of occurrences of the symbol a: Any accepting computation has to transition from state S to state A at some point. This is only possible by reading one of the as of the input and each possible a results in a different computation.

paths for an input word w. In a DFA any accepting computations are unique, which restricts us to counting functions $\Sigma^\star \to \{0, 1\}$.

By allowing multiple successors per symbol, we instead obtain a *non-deterministic finite automaton (NFA)*. Since we instead define our NFAs in a different (but mostly equivalent) way via so called $\mathbb{N}$-weighted automata, we postpone a formal definition to Section 4.2. A computation of an NFA M on some word w is a path in M labeled with w from some initial state to some other state. It is accepting if it ends in an accepting state and rejecting otherwise. Often there are now multiple computations of M on w, so M accepts w if *any* computation of M on w accepts. Similarly to DFAs, this allows an NFA M to recognize a language

$$L(M) := \{w \in \Sigma^\star \mid M \text{ accepts } w\}\,.$$

Note that both DFAs and NFAs recognize the same set of languages, the *regular languages*. To construct an equivalent DFA to some NFA M one can use the well known power-set construction where the DFA tracks inside its state space the set of all states any partial computation of M could be in.

The main power of NFAs, however, that we are interested in, comes with their ability to also represent counting functions: An NFA M computes a function $f : \Sigma^\star \to \mathbb{N}$, if the number of accepting computations of M on input w is exactly $f(w)$.

As one of the simplest examples, it can count the number of occurrences of some symbol $\sigma \in \Sigma$. We give the NFA computing $f : \{a, b\}^\star \to \mathbb{N}$ where $f(w)$ is number of occurrences of the symbol a in w in Figure 2.2.

There is an obvious limit to the functions that any NFA M can compute: The number of computations of M on some word of length n is always bounded by c^n where c is a constant only depending on M.

2.2.2 Turing Machines

Finite automata clearly are limited by the size of their memory (which is just their state space). To expand the power of our model and obtain a *(deterministic) Turing machine (DTM or just TM)* we add an infinite storage, a sequence of cells (called a *tape*), each containing a single symbol from a separate finite tape alphabet $\Gamma \supsetneq \Sigma$, together with a marker at one cell (called a *head*). Initially this tape contains the input word, surrounded on both sides by an infinite amount of some symbol $\square \in \Gamma \setminus \Sigma$ called an *empty*. At each step, the Turing machine reads the symbol at the head and uses its state space to determine what symbol it should write at the head, which state it should transition into and whether to leave the head where it is or move it one position to the left or right. If the Turing machine has no next state, it terminates and either accepts or rejects, depending on the state it terminated in.

Similarly to finite automata, a DTM decides a language

$$L(M) := \{w \in \Sigma^\star \mid M \text{ accepts } w\}.$$

If, instead, we want to use a DTM to compute an arbitrary function $\Sigma^\star \to (\Gamma \setminus \{\square\})^\star$ we can take the output of the DTM M on input w to be the content of the tape once M terminates (after removing all $\square$ symbols).

We can visualize Turing machines in a graphical way similarly to finite automata, however even describing a simple program this way quickly becomes unwieldy. Instead we will usually resort to describing the behavior of a TM using high level pseudocode.

We generalized deterministic finite automata to non-deterministic finite automata by allowing transitions to choose among multiple options. We can do the same with our deterministic Turing machines to obtain the *non-deterministic Turing machine (NTM)*. When talking about these options in a high level description, we say the NTM *non-deterministically guesses* or just *guesses*. For example given a graph $G = (V, E)$, an NTM could non-deterministically guess a subset $V' \subseteq V$ of the vertices. The NTM would then use multiple non-deterministic choices to branch each computation into $2^{|V|}$ computations: one for each subset $V' \subseteq V$.

As usual, an NTM M accepts a word w if any computation of M on w accepts, allowing M to decide a language

$$L(M) := \{w \in \Sigma^\star \mid M \text{ accepts } w\}.$$

On the other hand M computes a natural counting function $f : \Sigma^\star \to \mathbb{N}$, where $f(w)$ is the number of accepting computations of M on input w.

In the context of complexity classes we regularly constraint either the running time or the space usage of a deterministic or non-deterministic Turing machine M. On a fixed input w, the running time of M is the maximum number of steps until termination in any computation of M on w, while the space usage of M is the maximum number of cells on the tape the head has been on during any computation.

As a dependence on the input length, M is said to have running time $t(n)$ if it has a running time of at most $t(n)$ on any word of length n. Similarly, M has space usage $s(n)$ if it uses at most $s(n)$ space on any word of length n.

2.2.3 Real Word RAMs

Turing machines can only do discrete computations due to their dependence on a finite alphabet. In particular Turing machines are unable to store arbitrary real numbers.

The *real word random access machine (real word RAM)* was introduced by Erickson et al. [EHM24] and is an extension to the word RAM. In comparison to Turing machines a (real) word RAM consists of a list of instructions (similar to assembler languages) and memory partitioned into registers. The real word RAM itself is a model that has both real registers $R[i]$ and integer registers $W[i]$. The $R[i]$ contain arbitrary real numbers while the $W[i]$ contain a *word*[2], i.e. an integer between 0 and $2^w - 1$ for some fixed *word size* w. The two different types of registers allow for different operations. As such, the word registers $W[i]$ support

- arithmetic and bitwise Boolean operations (addition, subtraction, multiplication (including double precision), rounded division, remainder and bitwise nand),
- comparisons between word registers with conditional jumps, and, most importantly,
- indirect memory access to both types of registers, i.e. operations of the forms $W[W[i]] \leftarrow W[j]$ or $R[i] \leftarrow R[W[j]]$ and similar.

The real registers $R[i]$ are much more restricted and only support

- exact arithmetic operations (addition, subtraction, multiplication and division), and
- comparing a real register to 0 (both $= 0$ and > 0) for conditional jumps.

Moreover, we can cast word registers into real registers ($R[i] \leftarrow W[j]$), but there is no way to extract any information from a real register except by comparison to zero.

Note that this model is similar to the so-called BSS model of real computation [BSS89], but distinctly different by virtue of having both real and word registers.

To extend real word RAMs to non-deterministic real word RAMs, we add a non-deterministic guessing operation where the machine can fill any chosen register with

[2] We are well aware that we unfortunately have two different objects called "words" within this work. Once for strings and once for the range of integer registers as part of our word RAMs. Both have historically used the term "word" and we believe it might cause more confusion trying to rename them. We hope that it will always be clear from the context which of the two we mean.

an arbitrary value. In case of a word register this branches the computation into 2^w computations, while for real registers it branches into uncountably many computations, one for each real number. Acceptance on some input (a sequence of integers and a sequence of real numbers) is then as usual defined as the acceptance of any computation.

2.2.4 Circuits and Formulas

A *circuit* is a directed, acyclic graph where every node is labeled with either an operation, an input variable or a constant. The nodes are called *computation node*, *input node* or *constant node* respectively. The in-degree of each node is called its *arity*. Input and constant nodes always have arity 0, while computation node arities depend on the operation. If the operation is not commutative, then the order of the incoming edges matters. *Output nodes* are nodes with out-degree 0 and they have a fixed order among each other.

Depending on the type of circuit, the allowed operations for the computation nodes vary. For a *Boolean circuit* they can be the logical operations $\wedge$ (arity 2), $\vee$ (arity 2), and $\neg$ (arity 1), while for an *arithmetic* (sometimes also called *algebraic*) *circuit* they are the operations $+$ and $\cdot$ (both arity 2).

A circuit then computes values for all nodes inductively, starting from the input and constant nodes and for each computation node, labeled with some operation, applying said operation to the values of its parents to obtain its own value. The output of the circuit is then the tuple (or concatenation, whichever is convenient) of values at the output nodes in order.

A circuit is called a *formula* if no node has an out-degree bigger than one.

There are two primary complexity measures associated with circuits and formulas: size and depth. The *size* of a circuit is the number of nodes within the circuit, while the *depth* of a circuit is the length of (i.e. the number of edges on) the longest path from any input to any output node.

2.3 Reductions and Completeness

Given two languages $L, L' \subseteq \Sigma^\star$, we say that L is *polynomial time (many-one) reducible* to L' (written $L \leq_P L'$) if there is a polynomial time computable function $f : \Sigma^\star \to \Sigma^\star$, s.t. $x \in L \iff f(x) \in L'$. If we have both $L \leq_P L'$ and $L' \leq_P L$, we write $L \equiv_P L'$.

A complexity class $\mathcal{C}$ is said to be *closed under polynomial time reductions*, if, whenever $L \in \mathcal{C}$ and $L' \leq_P L$, then $L' \in \mathcal{C}$. A language $L \subseteq \Sigma^\star$ is said to be $\mathcal{C}$*-hard* (under polynomial time reductions), if we have $L' \leq_P L$ for all $L' \in \mathcal{C}$. Moreover, L is said to be $\mathcal{C}$*-complete* (under polynomial time reductions) if it is $\mathcal{C}$-hard and additionally $L \in \mathcal{C}$.

Intuitively, if some L is $\mathcal{C}$-hard, then it is at least as difficult as all languages in $\mathcal{C}$ and if it is $\mathcal{C}$-complete, it is among the hardest problems in $\mathcal{C}$.

Similarly we also define closure of complexity classes under other types of reductions, as well as hardness and completeness. For example we will see so called *p-projections*, as a rather weak type of reductions between families of polynomials in Section 2.5.4.

2.4 Oracles

"What if some model could efficiently answer question X?" This natural question can be answered by adding a separate oracle tape to a Turing machines and a special state that, whenever reached, takes the content of the oracle tape, computes the answer to question X on it and then replaces the content of the oracle tape with the answer in a single step. For a (real) word RAM we can similarly add some additional special oracle registers and an oracle instruction with the same behavior.

For a complexity class $\mathcal{C}$ defined via one of these models and a language L, we write $\mathcal{C}^L$ for the set of all languages decidable in the same way, but the model is augmented with an L-oracle. Usually we don't care about the precise problem used as an oracle and only about its complexity. We then write $\mathcal{C}^\mathcal{D}$ for a complexity class $\mathcal{D}$ to mean $\bigcup_{L\in\mathcal{D}} \mathcal{C}^L$. Generally, if $\mathcal{C}$ at least allows for polynomial time computations, this is then equivalent to $\mathcal{C}^L$ for any $\mathcal{D}$-complete (under polynomial time reductions) problem L.

2.5 Complexity Classes

For the readers convenience we give a quick overview over the complexity classes we use throughout this work.

2.5.1 Classical Complexity Classes

All the classes mentioned in this section are closed under polynomial time reductions. Thus, when we talk about hardness/completeness for these classes in this work, unless noted differently, we mean hardness/completeness under polynomial time reductions.

P: The set of all languages L that can be decided by a deterministic Turing machine in time polynomial in the input length. Generally considered to be the class of "efficiently solvable problems".

NP: The set of all languages L that can be decided by a non-deterministic Turing machine in time polynomial in the input length, i.e. $x \in L$ iff there is an accepting path on input x. Equivalently, it is the set of languages L, s.t. there is a (deterministically) polynomial time computable relation R and a polynomial p with

$$x \in L \Leftrightarrow \exists y \in \Sigma^{\leq p(|x|)} : R(x, y) .$$

Thus these are the class of "efficiently verifiable problems", also motivating the name $\exists\mathsf{P}$ for NP.

The classical NP-complete problem is SAT, the problem of deciding Boolean satisfiability.

co-NP: The co-class of NP, i.e. co-NP $= \{\overline{L} \mid L \in \mathsf{NP}\}$. Phrasing it in terms of non-deterministic Turing machines, a language L is in co-NP if there is a non-deterministic Turing machine M, s.t. $x \in L$ iff all paths of M on input x accept. Equivalently, it is the set of languages L, s.t. there is a (deterministically) polynomial time computable relation R and a polynomial p with

$$x \in L \Leftrightarrow \forall y \in \Sigma^{\leq p(|x|)} : R(x, y) .$$

Thus these are the class of "efficiently falsifiable problems", also motivating the name $\forall\mathsf{P}$ for co-NP.

PH: Extending the ideas of $\exists\mathsf{P}$ and $\forall\mathsf{P}$ further, we can also allow for multiple quantifiers. For example $\exists\forall\mathsf{P}$ is the set of languages L, s.t. there is a (deterministically) polynomial time computable relation R and a polynomial p with

$$x \in L \Leftrightarrow \exists y \in \Sigma^{\leq p(|x|)} \forall z \in \Sigma^{\leq p(|x|)} : R(x, y, z) .$$

The polynomial time hierarchy PH is then the union of all classes of this form with a finite number of quantifiers.

PP: The set of all languages L where a non-deterministic polynomial time Turing machine M exists, s.t. $x \in L$ iff at least half the paths of M on input x accept.

$\mathsf{NP}^{\mathsf{PP}}$: The set of all languages decidable by a non-deterministic polynomial time Turing machine, when provided with a PP oracle.

We have $\mathsf{NP}^{\mathsf{PP}} = \mathsf{NP}^{\#\mathsf{P}}$ (a definition of #P will follow in Section 2.5.2). The PP oracle can simulate the #P oracle by binary search.

The canonical complete problem for $\mathsf{NP}^{\mathsf{PP}}$ is E-MajSat: Given a Boolean formula ϕ in the variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, decide whether there is an assignment to the x_i, s.t. at least half the assignments to the y_i satisfy ϕ [LGM98].

PSPACE: The set of all languages L that can be decided by a deterministic Turing machine in polynomial space in the input length. Equivalently, it is the set of languages L decidable in parallel polynomial time (with an exponential number of processors) on a PRAM[3] [FW78].

[3]The PRAM is a parallel random access machine. See [FW78] for a formal definition.

The canonical PSPACE-complete problem is QBF: given a quantified Boolean formula $Q_1 x_i \, Q_2 x_2 \, \ldots \, Q_n x_n \, \phi$ with $Q_i \in \{\exists, \forall\}$, decide whether it is a true sentence.

NEXP: The set of all languages L that can be decided by a non-deterministic Turing machine in time exponential in the input length, i.e. $x \in L$ iff there is an accepting path on input x.

Often, when some problem is NP-complete, a succinct version of the same problem is NEXP-complete. For example, take the NP-complete problem SAT of deciding Boolean satisfiability: A formula ϕ with $n = 2^N$ nodes is succinctly encoded as a circuit $C : \{0,1\}^N \to \{0,1\}^M$. On input of some node x, $C(x)$ outputs

- The label of x, i.e. whether it is a an input or computation node and the corresponding variable or operation respectively.
- The children of x if applicable.

The succinct variant of SAT, which has as input only the circuit C is then NEXP-complete.

2.5.2 Counting Complexity Classes

FP: The set of all functions f that can be computed by a deterministic Turing machine in time polynomial in the input length. Not strictly a counting class and rather a general function complexity class, however, we can interpret binary outputs as counting.

#P: The set of all counting functions f where a non-deterministic polynomial time Turing machine M exists, s.t. on input x, exactly $f(x)$ paths of M accept. Equivalently, the set of all counting functions f, s.t. there is a (deterministically) polynomial time computable relation R and a polynomial p with

$$f(x) = \left|\{y \in \Sigma^{\leq p(|x|)} \mid R(x,y)\}\right| .$$

#FA: The set of all counting functions f where a non-deterministic finite automaton M exists, s.t. on input x, exactly $f(x)$ paths of M accept. In the literature also known as the set of recognizable series $\mathbb{N}^{\mathrm{rec}}\langle\langle\Sigma^\star\rangle\rangle$, but we use the name #FA to highlight the similarities to #P.

#P$^{\cdot\cdot}$: Similar to #P, however the main part of the input is given via an oracle (symbolized by the dotted circle), while the running time is only counted in terms of the classical input (usually a binary encoding of the instance size). Defined formally in Definition 5.6.

2.5.3 Search Problems

So far we have considered decision and counting problems. Search problems are of a different flavor: Instead of determining whether there is some solution, they require you to find a solution.

TFNP: The class of functions f of the following form:
Given an input x and polynomial-time relation R represented as an oracle O, output any y, s.t. $R(x, y)$.
Importantly, the relation R is syntactically guaranteed to have such a y for every choice of x.

PPA, PPAD, PPADS, PPP and PLS: These are all subclasses of TFNP, with different reasons for why the existence of y is guaranteed. Going into detail here is out of scope of this work and not required for understanding it. To the interested reader, we recommend the Complexity Zoo [AKH] for a quick overview and further references.

2.5.4 Algebraic Complexity Classes

These classes are no longer sets of languages or counting functions, but instead are sets of p-families as defined below.

Let $X = (X_1, X_2, \dots)$ be an infinite family of indeterminates over some field $\mathbb{F}$.

2.2 Definition. A sequence of polynomials $(f_n) \in \mathbb{F}[X]$ is called a *p-family* if for all n,

1. $f_n \in \mathbb{F}[X_1, \dots, X_{p(n)}]$ for some polynomially bounded function p and
2. $\deg f_n \leq q(n)$ for some polynomially bounded function q.

Let $f \in \mathbb{F}[X]$ be a polynomial and $s : X \to \mathbb{F}[X]$ be a mapping that maps indeterminates to polynomials. s can be extended in a unique way to an algebra endomorphism $\mathbb{F}[X] \to \mathbb{F}[X]$. We call s a *substitution*. (Think of the variables replaced by polynomials.)

2.3 Definition. 1. Let $f, g \in F[X]$. f is called a *projection* of g if there is a substitution $r : X \to X \cup F$ such that $f = r(g)$. We write $f \leq_P g$ in this case. (Since g is a polynomial, it only depends on a finite number of indeterminates. Therefore, we only need to specify a finite part of r.)

2. Let (f_n) and (g_n) be p-families. (f_n) is a *p-projection* of (g_n) if there is a polynomially bounded function $q : \mathbb{N} \to \mathbb{N}$ such that $f_n \leq_P g_{q(n)}$. We write $(f_n) \leq_P (g_n)$.

Projections are very simple reductions. Therefore, we can also use them to define hardness for "small" complexity classes like VP.

For any p-family (f_n), the *circuit complexity* of f_n is the size of the smallest arithmetic circuit computing f_n, as a function in n.

$\mathsf{VP}_{\mathbb{F}}$: Valiant's P. The set of all p-families (f_n) with polynomially bounded circuit complexity. Considered to be an algebraic equivalent to P.

$\mathsf{VNP}_{\mathbb{F}}$: Valiant's NP. The set of all p-families (f_n), s.t. there are polynomially bounded functions p and q and a sequence $(g_n) \in \mathsf{VP}_{\mathbb{F}}$ of polynomials in the variables $X_1, \ldots, X_{p(n)}$ and $Y_1, \ldots, Y_{q(n)}$ such that

$$f_n = \sum_{e \in \{0,1\}^{q(n)}} g_n(X_1, \ldots, X_{p(n)}, e_1, \ldots, e_{q(n)}) \,.$$

One can think of the X-variables representing the input and the Y-variables the witness. With this interpretation, VNP is more like $\#\mathsf{P}$. Also considered to be an algebraic equivalent to NP.

Whenever $\mathbb{F}$ has a characteristic different from 2, then the permanent

$$\mathrm{per}_n = \sum_{\sigma \in \mathfrak{S}_n} X_{1,\sigma(1)} \cdots X_{n,\sigma(n)}$$

is $\mathsf{VNP}_{\mathbb{F}}$-complete under p-projections.

In the above example, the indeterminates have two indices instead of one. Of course we could write per_n as a polynomial in $X_1, X_2, \ldots$ by using a bijection between $\mathbb{N}^2$ and $\mathbb{N}$. However, we prefer the natural naming of the variables (and will do so with other polynomials).

2.5.5 The (Existential) Theory of the Reals

The theory of the reals forms a hierarchy, similar to the polynomial time hierarchy PH, see [SS24]. While technically classical complexity classes, due to our focus on (variations of) the theory of the reals in Chapter 6 and some applications in Chapters 8 and 9, we go into a bit more detail about this hierarchy here, even including some common results we will need.

We will start by describing the first level of this hierarchy and afterwards continue by showing how to extend this definition to the remaining levels.

First Level

The existential theory of the reals (ETR) is the set of true sentences of the form

$$\exists x_1 \dots \exists x_n \ \varphi(x_1, \dots, x_n), \tag{2.4}$$

where φ is a quantifier-free Boolean formula over the basis $\{\vee, \wedge, \neg\}$ and a signature consisting of the constants 0 and 1, the functional symbols $+$ and $\cdot$, and the relational symbols $<$, $\leq$, and $=$. The sentence is interpreted over the real numbers in the standard way. The theory forms its own complexity class $\exists\mathbb{R}$ which is defined as the closure of ETR under polynomial-time many-one reductions. Many natural problems have been shown to be complete for ETR, for instance the computation of Nash equilibria [SŠ17], the famous art gallery problem [AAM18], or training neural networks [Ber+23], just to mention a few. See the recent compendium [SCM24] for a complete overview.

It turns out that one can simplify the form of an ETR-instance. We can get rid of the relations $<$ and $\leq$ and it is sufficient to consider only Boolean conjunctions. More precisely, the following problem is $\exists\mathbb{R}$-complete: Given polynomials $p_1, \dots, p_m$ in variables $x_1, \dots, x_n$, decide whether there is a $\xi \in \mathbb{R}^n$ such that

$$p_1(\xi) = \dots = p_m(\xi) = 0. \tag{2.5}$$

By Tseitin's trick, we can assume that all polynomials are of one of the forms

$$ab - c, \ \ a + b - c, \ \ a - b, \ \ a - 1, \ \ a \tag{2.6}$$

and all variables in each of the polynomials are distinct. Note that all polynomials in (2.6) have degree at most two. Therefore, this problem is also called the feasibility problem of quadratic equations QUAD.

2.7 Lemma (Variation of the proof from [SŠ17])**.** QUAD *is* $\exists\mathbb{R}$*-complete.*

Proof. Let $\exists x_1 \dots \exists x_n \ \varphi(x_1, \dots, x_n)$ be a sentence with φ being a quantifier free Boolean formula as above. W.l.o.g. assume that φ is monotone relative to all the occurring relations, i.e. that φ contains no negations $\neg$. We can always achieve this by pushing negations inwards using De Morgan's law and then absorbing them into the relations. If this creates a relation $f \neq g$, replace it by $(f < g) \vee (f > g)$. First, replace all occurrences of relations $f \leq g$ by $(f < g) \vee (f = g)$. Next, we show how to eliminate the relational symbol $<$. Let $f < g$ be one such relation. Introduce a fresh existentially quantified slack variable s and replace this relation by $f + s^2 = g$. This is equivalent to $f < g$ whenever $s \neq 0$, which we enforce by additionally adding the requirement $s \cdot s' = 1$ for another fresh variable s'.

Repeating this procedure we obtain a quantifier free Boolean formula φ' over the basis $\{\vee, \wedge\}$, only containing equality relations. Remains to show how to transform φ' into a conjunction of polynomials as in (2.6) using Tseitin's trick: We proceed inductively over

the structure of φ', adding a fresh variable t_g for each gate g, computing the value of the subtree of φ' rooted at g whenever g computes an arithmetic expression, otherwise $t_g = 0$ iff the value computed at g is true.

Constant 0 gate: Add the equation $t_g = 0$.

Constant 1 gate: Add the equation $t_g - 1 = 0$.

Variable gate with variable x_i: Add the equation $x_i - t_g = 0$.

Addition gate with children a, b: Add the equation $t_a + t_b - t_g = 0$.

Multiplication gate with children a, b: Add the equation $t_a \cdot t_b - t_g = 0$.

Equality gate with children a, b: Add the equation $t_g + t_a - t_b = 0$.

Conjunction gate with children a, b: Add the equation $t_a^2 + t_b^2 - t_g = 0$.

Disjunction gate with children a, b: Add the equation $t_a \cdot t_b - t_g = 0$.

Finally, for the root gate r, add the equation $t_r = 0$.

It is easy to verify that each of these constructions fulfills the inductive invariant.

Note that the replacements for conjunctions are not of the form (2.6) yet and contain some variables twice. However, applying the above transformation a second time will ensure those requirements. □

Universal Quantification. If, instead of considering existentially quantified true sentences, we consider universally quantified true sentences of the form

$$\forall x_1 \dots \forall x_n \, \varphi(x_1, \dots, x_n), \tag{2.8}$$

where φ is again a quantifier-free Boolean formula, and form the closure under polynomial-time many-one reductions, we obtain the complexity class $\forall\mathbb{R}$. Using De Morgan's law, it is easy to see the well-known fact that $\forall\mathbb{R} = \text{co-}\exists\mathbb{R}$, i.e. it is the complement class of $\exists\mathbb{R}$.

Higher Levels

By allowing a finite amount of quantifier alternations, we obtain a whole hierarchy, comparable to the well-known polynomial time hierarchy, see [SS24]. For example when considering true sentences of the form $\exists x_1, \dots, x_n \, \forall y_1, \dots, y_m \; \varphi(x_1, \dots, x_n, y_1, \dots, y_m)$ and forming the closure under polynomial-time many one reductions we obtain the class $\exists\forall\mathbb{R}$. Similarly, we obtain the classes $\forall\exists\mathbb{R}$, $\exists\forall\exists\mathbb{R}$, $\forall\exists\forall\mathbb{R}$, ...

Complexity

It is easy to see that quantification over real variables can be used to simulate quantification over Boolean variables by adding the constraint $x(x-1)=0$. This way we can convert 3SAT-formulas to ETR-formulas, proving the well-known containment $\mathsf{NP} \subseteq \exists\mathbb{R}$.

An upper bound is given by Renegar's celebrated result about quantifier elimination:

2.9 Lemma (Theorem 1.1 in [Ren92]). *Let $p_1, \dots, p_m$ be polynomials in n variables $x_1, \dots, x_n$ and of degree at most $d \geq 2$ where all coefficients are given as integers of bit length L. Additionally let $\circ_1, \dots, \circ_m \in \{>, <, =\}$ be comparison operators, let P be a Boolean function with m inputs and let $Q_1, \dots, Q_n \in \{\exists, \forall\}$ be quantifiers with ω blocks of quantifiers with sizes $n_1, \dots, n_\omega$[4]. Then there is an algorithm deciding whether $Q_1 x_1\, Q_2 x_2 \dots Q_n x_n\, P(p_1(x_1, \dots, x_n) \circ_1 0, \dots, p_m(x_1, \dots, x_n) \circ_m 0) = 1$ is a true sentence (when all variables are quantified over the real numbers) in time*

$$L \log(L) \log\log(L)\, (md)^{2^{O(\omega)} \prod_i n_i}$$

using $(md)^{O(n)}$ evaluations of P.

Furthermore there is a parallel algorithm deciding the same in time

$$\log(L) \left(2^\omega \left(\prod_i n_i\right) \log(md)\right)^{O(1)} + Time(P, k)$$

using $L^2 (md)^{2^{O(\omega)} \prod_i n_i}$ processors and requiring $(md)^{O(\sum_i n_i)}$ evaluations of P. Here $Time(P, k)$ is the parallel time needed for the evaluations of P when using $k(md)^{O(\sum_i n_i)}$ processors (for any $k \geq 1$).

[4] A block of quantifiers of size n_i is a sequence of n_i consecutive quantifiers with the same quantification. For example $\exists\exists\forall\forall\exists$ has $\omega = 3$, $n_1 = 2$, $n_2 = 3$ and $n_3 = 1$.

In particular, whenever the number ω of quantifier blocks is constant, this is a parallel algorithm running in polynomial time, implying decidability in PSPACE [FW78] and leading to the inclusions

$$\mathsf{NP} \subseteq \exists\mathbb{R} \subseteq \mathsf{PSPACE} \qquad \text{and} \qquad \mathsf{co\text{-}NP} \subseteq \forall\mathbb{R} \subseteq \mathsf{PSPACE}\,. \tag{2.10}$$

While all these inclusions are believed to be strict, it is unknown for all of them.

Related Complexity Classes

There are many complexity classes related to the theory of the reals. All of these are covered and defined in more detail in Chapter 6.

succ-$\exists\mathbb{R}$: The succinctly encoded variant of $\exists\mathbb{R}$, similar to how NEXP can be viewed as a succinctly encoded variant of NP. Recently introduced in [ZBL23].

succ-$\exists\mathbb{R}_{\text{poly}}$: Defined in the same way as succ-$\exists\mathbb{R}$, however the succinctly encoded ETR formulas are only allowed to use polynomially many different variables. This class is equal to PSPACE.

$\exists\mathbb{R}^{\Sigma}$: Defined as the closure of a variant of ETR under polynomial time reductions where we augment ETR by adding a unary summation symbol $\sum_{e=0}^{1}$.

$\exists\mathbb{R}^{\Pi}$: Defined as the closure of a variant of ETR under polynomial time reductions where we augment ETR by adding a unary product symbol $\prod_{e=0}^{1}$. This class is equal to PSPACE.

$\mathsf{NP}_{\mathsf{real}}$: The set of all languages L that can be decided by a non-deterministic word RAM in time polynomial in the input length. This class is equal to $\exists\mathbb{R}$ and gives a machine model characterization rather than a definition via a complete problem [EHM24].

$\mathsf{NEXP}_{\mathsf{real}}$: The set of all languages L that can be decided by a non-deterministic word RAM in time exponential in the input length. This class is equal to succ-$\exists\mathbb{R}$ and gives a machine model characterization rather than a definition via a complete problem.

$\mathsf{NP}_{\mathsf{real}}^{\mathsf{VNP}_{\mathbb{R}}}$: The set of all languages L that can be decided by a non-deterministic word RAM in time polynomial in the input length when given access to a $\mathsf{VNP}_{\mathbb{R}}$ oracle. This class is equal to $\exists\mathbb{R}^{\Sigma}$ and gives a machine model characterization rather than a definition via a complete problem.

2.6 Promise Problems

In a classical decision problem L, we are given an input and we have to decide whether $x \in L$ (the so-called yes-instances) or $x \notin L$ (the no-instances). For instance, in the classical SAT problem, we are given a Boolean formula F in CNF. The yes-instances are the satisfiable formulas and the no-instances are the unsatisfiable one.

Promise problems have a third type of instances, the so-called do-not-care-instances. On these instances, an algorithm can do what it wants and give any output. For instance, consider the problem SAT^{++}, where we ask the question of whether a satisfiable formula in CNF has another satisfying assignment. The yes-instances are all F with at least two satisfying assignments, the no-instances are all F with exactly one satisfying assignment, and the do-not-care-instances are all unsatisfiable F. An algorithm solving SAT^{++} has to output "yes" on every F with at least two satisfying assignments and "no" on every F with exactly one satisfying assignment. On unsatisfiable formulas, it can output

whatever it wants. Note that every classical decision problem is also a promise problem with the do-not-care-instances being the empty set.

We can also define many-one reductions for promise problems: A function $f : \{0,1\}^* \to \{0,1\}^*$ is called a many-one reduction from a promise problem L to another promise problem L', if f maps yes-instances of L to yes-instances of L' and no-instances of L to no-instances of L'. f can map do-not-care-instances of L to any instance of L'.

One can show that SAT^{++} is NP-hard, since we can reduce SAT to it. This reduction maps the unsatisfiable formulas (no-instances of SAT) to formulas with one satisfying assignment (no-instances of SAT^{++}) and satisfiable formulas (yes-instances of SAT) to formulas with two or more satisfying assignments (yes-instances of SAT^{++}). Since SAT is a classical decision problem, there are no do-not-care-instances. SAT^{++} is, however, not contained in NP for formal reasons, because NP only contains classical decision problems.

2.7 (Semi)algebraic Sets

Directly related to the existential theory of the reals, we give a brief introduction to (semi)algebraic sets and discuss the notations important for this work. For details and proofs, we refer to the book [BPR03].

An *algebraic set* (sometimes also called an *algebraic variety*[5]) in $\mathbb{R}^n$ is a set of the form $\{(x_1,\dots,x_n) \mid f_1(x_1,\dots,x_n) = \dots = f_m(x_1,\dots,x_n) = 0\}$ where $f_1,\dots,f_m$ are real polynomials in n variables. It is called *irreducible* if it cannot be written as the union of two smaller algebraic sets.

[5] some authors only call irreducible algebraic sets varieties, however we will use algebraic sets and algebraic varieties as equivalent throughout this work.

A *semialgebraic set* in $\mathbb{R}^n$ is a finite Boolean combination (finite number of unions and intersections) of sets of the form $\{(x_1,\dots,x_n) \mid f(x_1,\dots,x_n) > 0\}$ and $\{(x_1,\dots,x_n) \mid g(x_1,\dots,x_n) \geq 0\}$. Here f and g are real polynomials in n variables.

A *semialgebraic function* is a function $\mathbb{R}^n \to \mathbb{R}^{n'}$ with a semialgebraic graph, that is, the set of all $\{(x, f(x)) \mid x \in \mathbb{R}^n\}$ is a semialgebraic set.

From this definition of semialgebraic sets, it is easy to see that semialgebraic sets are the solutions of ETR-instances, that is, all $(x_1,\dots,x_n)$ satisfying $\varphi(x_1,\dots,x_n)$ in (2.4) form a semialgebraic set. From Tarski's theorem (see [BPR03]), it follows that semialgebraic sets allow quantifier elimination, that is, all $(x_1,\dots,x_n)$ satisfying

$$\exists y_1 \dots \exists y_t \, \psi(x_1,\dots,x_n,y_1,\dots,y_t)$$

form a semialgebraic set, where ψ (like φ) is a quantifier-free Boolean formula over the basis $\{\vee,\wedge,\neg\}$ and a signature consisting of the constants 0 and 1, the functional symbols $+$ and $\cdot$, and the relational symbols $<$, $\leq$, and $=$. ψ depends on two sets of variables. It is clear that all $(x_1,\dots,x_n,y_1,\dots,y_t)$ satisfying ψ form a semialgebraic set. Tarski's theorem tells us that we still get a semialgebraic set when we are projecting the

$y_1, \ldots, y_t$ away using the existential quantifiers. Combined with the $\exists\mathbb{R}$-completeness of QUAD, the semialgebraic sets are precisely projections of algebraic sets. These facts are used frequently in our proofs.

2.11 Definition. A semialgebraic set S has dimension d if there exists a d-dimensional coordinate subspace such that the image of S under the canonical projection onto this subspace has a nonempty interior and there is no such subspace of dimension $d+1$.

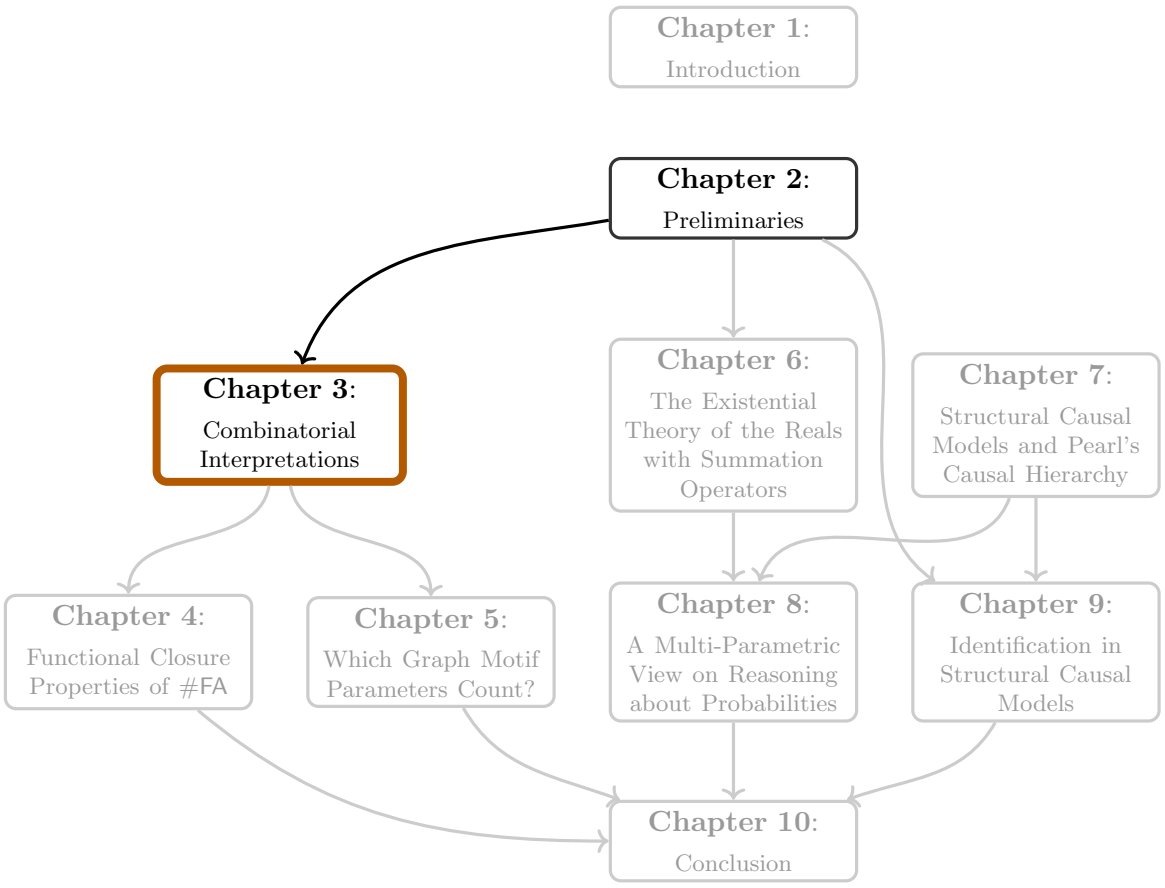
Above, by a d-dimensional coordinate subspace, we mean the subspace of $\mathbb{R}^n$ of all points x such that $x_i = 0$ for $i \notin I$ for some subset $I \subseteq \{1, \ldots, n\}$ and $|I| = d$.

2.12 Definition. An algebraic set S has dimension d if there is a chain $\emptyset \subseteq S_0 \subseteq S_1 \subseteq S_d$ of irreducible algebraic subsets of S and there is no larger d with this property.

The notions of dimensions for algebraic and semialgebraic sets are closely related:

2.13 Theorem (see e.g. [BCR13])**.** *Let S be a semialgebraic set. Then its dimension as a semialgebraic set (as in Definition 2.11) equals the dimension of its Zariski closure as an algebraic set.*

The *Zariski topology* has precisely the algebraic sets as its closed sets.



Combinatorial Interpretations

3.1 Positivity in Combinatorics

A fundamental task in combinatorics is to assign combinatorial meanings to quantities: For example, the binomial coefficient $\binom{n}{k}$ has the combinatorial interpretation of counting k-element subsets of $\{1, \dots, n\}$, which may not be obvious from the algebraic formula $\frac{n!}{k!(n-k)!}$. In fact, this is a *nonnegative* combinatorial interpretation, i.e., the binomial coefficient is equal to the cardinality of a naturally defined set. This is stronger than just having a *signed* combinatorial interpretation, such as for example the determinant of zero-one matrices $A = (a_{i,j})_{1 \leq i,j \leq n}$, which can be expressed as the difference of the number of even versus odd permutations π with $a_{i,\pi(i)} = 1$ for all $1 \leq i \leq n$. In the following, when we speak of combinatorial interpretations, we mean nonnegative ones.

In algebraic combinatorics, there are several nonnegative integer quantities for which finding such a combinatorial interpretation is a major open problem, e.g., problems 9, 10, 11, and 12 in [Sta00]. This begs the question whether these quantities actually have combinatorial interpretations, but without a formal definition of this notion, their existence is hard to rule out. Indeed, Stanley [Sta11, Ch. 1] writes that "only through experience does one develop an idea of what is meant by a "determination" of a counting function".

3.2 Formalized Approaches

3.2.1 Counting with Turing Machines

In many settings however, combinatorial interpretations of a nonnegative quantity can be defined (or at least subsumed) formally in terms of containment of the quantity in the counting complexity class #P, see [IP22; Pak23; Pan24]. This means that the quantity can be interpreted as counting accepting paths of some nondeterministic polynomial-time Turing machine. While "containment in #P" is a coarse over-approximation of what some mathematicians would consider "combinatorial interpretations" and even fails to include some operations like repeated power sets, this working definition is formal, robust, and arguably natural. Most importantly, it enables us to prove that some very natural quantities indeed do *not* have a combinatorial interpretation (or at the very least cannot have one without resorting to operations that #P is not closed under).

For example, if PH does not collapse, then the squares of the symmetric group characters are not in #P and hence do not have a combinatorial interpretation, see [IPP23]. Other fruitful separations from #P are proved in [IP22], where the counting versions of several TFNP search problems are studied: For example, given oracle access to an exponentially large graph with degree at most two (i.e., a disjoint union of isolated vertices, paths, and cycles), and given the end of a path, it follows that there is at least one other end of a path. But the task of finding a combinatorial interpretation for the nonnegative quantity "number of other path endpoints -1" fails due to the oracle separation in [IP22].

3.2.2 Local Combinatorial Interpretations

The previous approach of characterizing combinatorial interpretations as precisely the problems in #P has the problem of both under- and over-approximating what could be reasonably considered as "combinatorial": One one hand, for example, the determinant is computable in polynomial time and thus trivially in #P, yet the only known representation close to a combinatorial one is as a difference between two combinatorial functions: one for the even permutations and one for the odd permutations. On the other hand, the power set operation cannot be represented in #P: If an NTM takes time $p(|x|)$ and branches by at most a factor of c each step, the number of accepting paths is bounded by $c^{p(|x|)}$. Even applying the power set operation twice, however, already maps a set S to a set of size $2^{2^{|S|}}$, making a polynomial running time impossible.

Both of these counterexamples have one thing in common: they are inherently global. Even changing the input just slightly vastly changes the output. This motivates looking only at *local combinatorial interpretations* where we explicitly forbid global properties. Formally we achieve this by giving the NTM only oracle access to the input, while the classical input only contains the "input size", encoded in binary. The running time of such a machine is, however, still counted in terms of the classical input, thus making it impossible to read the full input in polynomial time. Each computation of the NTM can (and often should) still look at different parts of the inputs, but each individual computation can only look at a poly-logarithmic chunk of the input. This results in the type-2 complexity class $\#\mathsf{P}^{\circlearrowleft}$, defined in detail in Definition 5.6.

Going back to the example of (the absolute value of) a determinant of a zero-one matrix. An NTM where every computation can only look at a poly-logarithmic number of entries of the matrix is unable to distinguish between the identity matrix and the identity matrix where just a single entry on the diagonal is set to zero, even though the first one has determinant 1 and the latter has determinant 0: For the sake of contradiction, assume there is some computation that accepts the identity matrix. Then there are some entries on the diagonal this computation has never observed. Changing any of them to zero does not impact this computation, but the NTM is no longer allowed to have any accepting computations, a contradiction, proving that the determinant indeed does not have a local combinatorial interpretation.

While this work is the first time the concept of a local combinatorial interpretation is introduced explicitly, many of the results by Ikenmeyer and Pak [IP22] already implicitly use this class. They were proven for *relativized* #P, i.e. essentially $\#\mathsf{P}^{\circlearrowleft}$. Additionally, Hertrampf et al. [HVW95] have proven that the multi-variate closure properties of $\#\mathsf{P}^{\circlearrowleft}$ are precisely all functions that can be formed from addition, multiplication and binomial coefficients with fixed lower index, something that is conjectured for #P as well, but proving this would directly imply $\mathsf{P} \neq \mathsf{NP}$ [IP22], perhaps the most famous open problem in computer science.

By not being able to observe the whole input at once there is a new problem though:

An NTM is now unable to verify that the input even has the correct format. For example, when the input is supposed to be a tree, given as an oracle via the edge relation, connectedness and the lack of cycles are both global properties. There are two options here:

Syntactical Fixes: As is common in search classes like TFNP, one can choose a different encoding for the input that guarantees that any possible input is valid. In the example of our trees, giving the tree as an oracle representing the parent relation and requiring the parent of each vertex to have a smaller id than itself (defaulting to the root vertex with id 0 otherwise) always encodes a tree. This is also the primary method used in [IP22], but it is not clear that such a fix is always possible.

Promise Problems: The other, and in the eyes of the author the more natural, option is to consider the promise variant Pr-#P$^{\circlearrowleft}$ (see Section 2.6 for an introduction to promise problems). Here we only require the NTM to behave correctly when the oracle actually encodes a valid instance. This is our method of choice throughout this work.

In Chapter 5 we use this definition to show which motif parameters[1] allow for local combinatorial interpretations.

[1] Motif parameters are linear combinations of subobject counts, for example counting the number of 3-cliques plus the number of induced 4-cycles in a graph. See Section 5.2.3 for formal definitions.

3.3 Functional Closure Properties

We say a function $\varphi : \mathbb{N} \to \mathbb{N}$ is a *univariate functional closure property* of a complexity class $\mathcal{C}$ if $\varphi(\mathcal{C}) \subseteq \mathcal{C}$, i.e. if for every function $f \in \mathcal{C}$ the composition $\varphi \circ f$ is also in $\mathcal{C}$. A multivariate function $\varphi : \mathbb{N}^m \to \mathbb{N}$ is then a *functional closure property* of $\mathcal{C}$ if $\varphi(\mathcal{C}^m) \subseteq \mathcal{C}$, i.e. if for every m-tuple of functions $f_1, \ldots, f_m \in \mathcal{C}$ the composition $\varphi \circ (f_1, \ldots, f_m)$ is also in $\mathcal{C}$.

Functional closure properties can be studied for many different counting machine models (also for example with different types of oracle access) and different types of input sets. The first study of this type was done for nondeterministic polynomial-time Turing machines, i.e., the class #P, see [HVW95], [Bei97], and the recent [IP22]. The papers mentioned above prove that the relativizing multivariate polynomial closure properties are exactly those polynomials that have nonnegative integers in their expansion over the binomial basis, see [IP22]. A functional closure property $\varphi : \mathbb{N}^m \to \mathbb{N}$ of #P is *relativizing* if φ is a closure property for all #P^A, where $A \subseteq \Sigma^*$ is some oracle.

Functional closure properties can be used directly to construct combinatorial proofs of equalities and inequalities. For example, Fermat's little theorem states that p divides $a^p - a$. The quantity $\frac{1}{p}(a^p - a)$ has a combinatorial interpretation, which can be deduced from the fact that $\frac{1}{p}((f_1)^p - f_1)$ is a univariate functional closure property of #P, see [IP22, Prop. 7.3.1], which coincides with the original proof [Pet72], see also [Pak23, eq. (5)]). On the other hand, if a function is not a functional closure property, then this means in a very strong sense that there is no combinatorial interpretation for the

quantity it describes. For example, the Hadamard inequality ([HLP52, §2.13], [BB61, §2.11], [IP22, eq. (2)]) states that

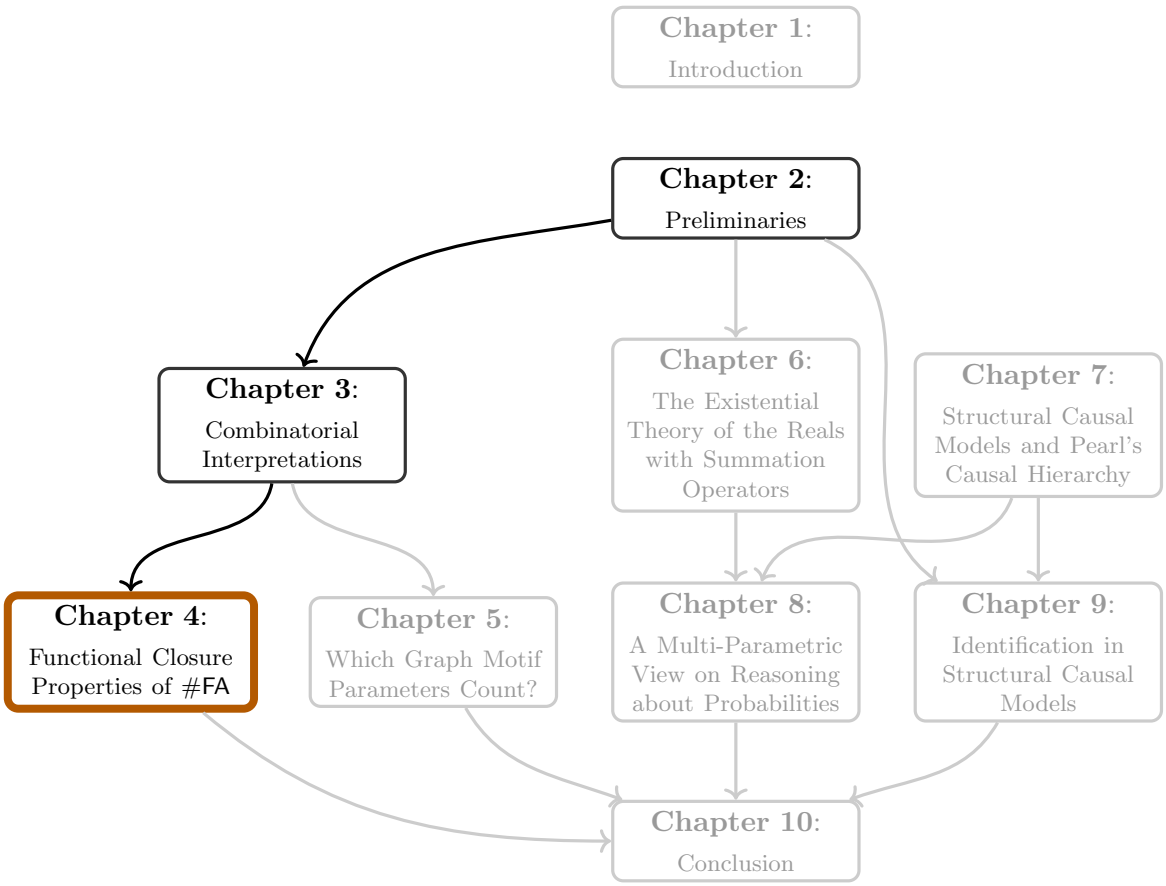
$$\det \begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{d1} & \cdots & a_{dd} \end{pmatrix}^2 \leq \prod_{i=1}^{d} (a_{i1}^2 + \cdots + a_{id}^2).$$

One could try to prove this by finding a combinatorial interpretation of the difference $\mathcal{H} \geq 0$ of the right-hand side and the left-hand side, but even for $d = 3$ we have that

$$\varphi(f_1, \ldots, f_9) \;=\; (f_1^2 + f_2^2 + f_3^3) \cdot (f_4^2 + f_5^2 + f_6^3) \cdot (f_7^2 + f_8^2 + f_9^3) - \det \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix}$$

is not a 9-variate relativizing functional closure property of #P, see [IP22, §7.2]. In particular, if the function is not a closure property of #P, then there are instantiations $f_1, \ldots, f_9 \in \#\mathsf{P}$ such that $\varphi(f_1, \ldots, f_9)$ is not in #P, whereas a combinatorial interpretation of $\mathcal{H}$ should yield $\varphi(f_1, \ldots, f_9) \in \#\mathsf{P}$. This, however, does not rule out a more indirect combinatorial proof for the inequality: For example, for proving combinatorially that $(a-1)^2 \geq 0$ one could try to interpret the quantity $(a-1)^2$ combinatorially, but $(f_1-1)^2$ is not a relativizing closure property of #P. However, $f_1 \cdot (f_1-1)^2 = 6\binom{f_1}{3} + 2\binom{f_1}{2}$ is a relativizing closure property of #P (see [IP22, §2.4]). There is an obvious combinatorial interpretation of $6\binom{a}{3} + 2\binom{a}{2}$ as counting size 2 and 3 subsets with multiplicity 6 and 2, respectively. Hence this gives an indirect combinatorial proof for the inequality $(a-1)^2 \geq 0$ by providing a combinatorial interpretation for $a(a-1)^2$.

We extend this line of research in Chapter 4, by determining the functional closure properties of #FA, i.e. counting the number of accepting paths of a nondeterministic finite automaton.



Functional Closure Properties of #FA

4.1 Finite $\mathbb{N}$-Weighted Automata and Functional Closure Properties

Let Σ be a finite set, for example $\Sigma = \{0, 1\}$. A finite $\mathbb{N}$-weighted automaton with all weights 1 is a nondeterministic finite automaton that on input $w \in \Sigma^\star$ outputs the number of accepting computation paths on input w, instead of just outputting whether or not an accepting computation path exists, see Def. 4.1 for the formal definition[1]. While every nondeterministic finite automaton determines a subset of $\Sigma^\star$, a finite $\mathbb{N}$-weighted automaton computes a function $\Sigma^\star \to \mathbb{N}$. A function $f : \Sigma^\star \to \mathbb{N}$ can be presented as the series $\sum_{w\in\Sigma^\star} f(w)w$, and the set of series is denoted by $\mathbb{N}\langle\langle\Sigma^\star\rangle\rangle$ in the automata literature, see e.g. [DK21][2]. The natural way of adding two functions $\Sigma^\star \to \mathbb{N}$ and adding two series in $\mathbb{N}\langle\langle\Sigma^\star\rangle\rangle$ coincides, but in both presentations we have a natural way of taking the product, and those do not coincide:

1. Pointwise product of functions $\Sigma^\star \to \mathbb{N}$. This is called the Hadamard product.
2. Convolution of series, called the Cauchy product.

A series f is called *recognizable* if there is a finite $\mathbb{N}$-weighted automaton that computes f. The set of recognizable series is denoted by $\mathbb{N}^{\mathrm{rec}}\langle\langle\Sigma^\star\rangle\rangle$ in [DK21], but we denote it by #FA, to emphasise that we undertake a study similar to #P in [HVW95, Thm 3.13], [Bei97, Thm 6], and recently [IP22], but instead of polynomial-time Turing machines we study finite automata. The Kleene-Schützenberger theorem states that #FA is the smallest set that contains all support 1 series and is closed under sums, Cauchy products, and Kleene-iterations (whenever well-defined, a Kleene-iteration is the sum of all Cauchy powers), see [DK21, §4], but we will not need this insight.

In this paper we study the functional closure properties[3] of #FA. A function $\varphi : \mathbb{N}^m \to \mathbb{N}$ is called a *functional closure property* of #FA if for all $f_1 \in$ #FA, $f_2 \in$ #FA, ..., $f_m \in$ #FA we have that $\varphi(f_1, \ldots, f_m) \in$ #FA. By $\varphi(f_1, \ldots, f_m)$ we mean the function that on input $w \in \Sigma^\star$ outputs $\varphi(f_1(w), \ldots, f_m(w))$.

Classically, one of the simplest functional closure properties of #FA is $\varphi : \mathbb{N}^2 \to \mathbb{N}$, $\varphi(f_1, f_2) = f_1 + f_2$. This is a functional closure property of #FA, because given $f_1 \in$ #FA and $f_2 \in$ #FA, we can show $\varphi(f_1, f_2) = f_1 + f_2 \in$ #FA by an easy construction: The new NFA consists of a copy of the NFA for f_1 and a copy of the NFA for f_2, and makes an initial nondeterministic choice as to which NFA to run, see Lemma 4.4 for the details.

Another classical simple functional closure property of #FA is $\varphi : \mathbb{N}^2 \to \mathbb{N}$, $\varphi(f_1, f_2) = f_1 \cdot f_2$. This corresponds to the Hadamard product. This is a functional closure property of #FA, because given $f_1 \in$ #FA and $f_2 \in$ #FA, we can show $\varphi(f_1, f_2) = f_1 \cdot f_2 \in$ #FA by the following construction: The new NFA consists of the product NFA of the NFAs for f_1 and f_2, and the accepting states correspond to pairs of accepting states, see Lemma 4.5 for the details.

[1] We use the equality of the number of accepting paths of an NFA and the output of the corresponding $\mathbb{N}$-weighted automaton, see [DK21, Exa. 2.2].

[2] A series with finite support is called a *polynomial*, but we will not be concerned with the support of series in this work. Instead, we use the term *polynomial* as it is used in commutative algebra, and we mean multivariate polynomials with rational coefficients.

[3] See [HVW95, Sec. 1] for the naming *functional closure property*. A different reasonable name would be *pointwise closure property*.

The Cauchy product is also a product on the set #FA, but we explain now that the Cauchy product is not "functional", and hence it is out of scope for this type of studies. If $\varphi : \mathbb{N}^m \to \mathbb{N}$ is a functional closure property of #FA, then we can study the corresponding map $\widetilde{\varphi} : \underbrace{\#\mathsf{FA} \times \#\mathsf{FA} \times \cdots \times \#\mathsf{FA}}_{m \text{ times}} \to \#\mathsf{FA}$. Observe that if φ is a functional closure property of #FA, then, by definition, we have that for all pairs $(w, w') \in \Sigma^\star \times \Sigma^\star$:

$$(f_1(w), \ldots, f_m(w)) = (f_1(w'), \ldots, f_m(w')) \implies \widetilde{\varphi}(f_1, \ldots, f_m)(w) = \widetilde{\varphi}(f_1, \ldots, f_m)(w') \,.$$

Let $\zeta : \#\mathsf{FA} \to \#\mathsf{FA}$ denote the Cauchy square. We use the observation above to show that ζ is not equal to $\widetilde{\varphi}$ for any $\varphi : \mathbb{N} \to \mathbb{N}$. Let $m = 1$ and $f(w) = 1$ if $w = 1$, $f(w) = 0$ otherwise. Clearly, $f \in \#\mathsf{FA}$. Then $\zeta(f)(11) = 1$, and $\zeta(f)(w) = 0$ for all $w \neq 11$. In particular $\zeta(f)(0) \neq \zeta(f)(11)$, even though $f(0) = f(11)$. Hence, $\zeta \neq \widetilde{\varphi}$ for all $\varphi : \mathbb{N} \to \mathbb{N}$.

Numerous functional closure properties of #FA exist, for example the safe decrementation $\max\{0, f_1 - 1\}$, and the binomial coefficient $\binom{f_1}{2}$. But not all non-negative functions are functional closure properties of #FA, for example $(f_1 - f_2)^2$ is not, which can be shown using the Pumping Lemma. In this paper, we determine *all* functional closure properties of #FA, see Section 4.1.1 for the detailed statement.

4.1.1 Our Results

Let $n \operatorname{rem} p \in \{0, \ldots, p-1\}$ denote the smallest nonnegative r such that $n \equiv_p r$. A function $\varphi : \mathbb{N} \to \mathbb{N}$ is called *ultimately PORC* (Polynomial On Residue Classes[4]) if $\exists p, N \in \mathbb{N}$ and there exist polynomials $\varphi_0, \ldots, \varphi_{p-1} : \mathbb{N} \to \mathbb{Q}$ such that for every $n \geq N$ we have $\varphi(n) = \varphi_{n \operatorname{rem} p}(n)$ [5].

We first classify the univariate functional closure properties of #FA:

- **Theorem** (see Theorem 4.27). *A function $\varphi : \mathbb{N} \to \mathbb{N}$ is a functional closure property of #FA if and only if φ is an ultimately PORC function.*

More generally, we classify the multivariate functional closure properties of #FA:

- **Theorem** (see Theorem 4.28). *A function $\varphi : \mathbb{N}^m \to \mathbb{N}$ is a functional closure property of #FA if and only if φ can be written as a finite sum of finite products of univariate ultimately PORC functions.*

We analyze the special case of multivariate polynomials:

- **Theorem** (see Lemma 4.31). *A multivariate polynomial $\varphi : \mathbb{N}^m \to \mathbb{N}$ with rational coefficients is a functional closure property of #FA iff for every ψ that can be formed from φ by replacing any subset of variables – including the empty set – by constants from $\mathbb{N}$, then all dominating terms of ψ in the binomial basis have positive coefficients.*

[4] PORC functions are also known as quasipolynomials or pseudopolynomials, but we want to avoid those names for the potential confusion to quasipolynomial growth and pseudopolynomial running times.

[5] Note that each φ in this paper is defined on the natural numbers and maps to the natural numbers, which is a subtle restriction. For example, a univariate polynomial $\varphi : \mathbb{Q} \to \mathbb{Q}$ maps integers to integers if and only if its coefficients in the binomial basis are integers, see Section 4.2. However, non-negativity is not an algebraic property. Also note that for the case of φ being just a univariate polynomial, the corresponding linear recursive sequence can have negative entries in the matrix.

We lift this result to monotone graph varieties (and to more general sets, see Theorem 4.38), where we get exactly the desirable classification given by the vanishing ideal:

- **Theorem** (see Theorem 4.41). *Let S be a monotone graph variety and let $I = I(S)$ be its vanishing ideal. A multivariate polynomial $\varphi : S \to \mathbb{N}$ is a functional promise closure property of #FA with regard to S if and only if there exists $\psi \in I$ such that $\varphi + \psi$ is a multivariate functional closure property of #FA.*

4.2 Notation

The vector space of multivariate polynomials $\mathbb{Q}[f_1, \dots, f_m]$ in variables $f_1, \dots, f_m$ has a basis given by products of binomial coefficients: $\left\{ \prod_{i=1}^{m} \binom{f_i}{c_i} \right\}_{c_1,\dots,c_m}$, where each $c_i \in \mathbb{N}$. Here we used $\binom{x}{c} = \frac{1}{c!} x \cdot (x-1) \cdot \ldots \cdot (x-c+1)$ as a polynomial. This is called the *binomial basis*. A multivariate polynomial φ is called *integer valued* if $\varphi(\mathbb{Z}^m) \subseteq \mathbb{Z}$, which is equivalent to $\varphi(\mathbb{N}^m) \subseteq \mathbb{Z}$, and which is also equivalent to all coefficients in the binomial basis being integers, see for example [IP22, Prop. 4.2.1] for a short proof of this classical fact.

We now recall (see [DK21, Def. 2.1]) our main model of computation, the finite $\mathbb{N}$-weighted automaton, which we just call non-deterministic finite automaton (NFA) for brevity.

4.1 Definition. An NFA M is a tuple $(Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ where the set of states Q and the alphabet Σ are finite sets and $\mathrm{wt} : Q \times \Sigma \times Q \to \mathbb{N}$ is the weighted transition function, $\mathrm{in} : Q \to \mathbb{N}$ are the weighted initial states and $\mathrm{out} : Q \to \mathbb{N}$ are the weighted accepting states[6]. A *computation* P for a word $w = w_1 \dots w_n \in \Sigma^\star$ of length n is a sequence $q_0 q_1 \dots q_n \in Q^{n+1}$. It has *multiplicity* or *weight*[7] $\mathbf{w}(P) = \mathrm{in}(q_0) \cdot \prod_{i=1}^{n} \mathrm{wt}(q_{i-1}, w_i, q_i) \cdot \mathrm{out}(q_n)$ and *partial weight* $\underline{\mathbf{w}}(P) = \mathrm{in}(q_0) \cdot \prod_{i=1}^{n} \mathrm{wt}(q_{i-1}, w_i, q_i)$. We say that M *computes* $f : \Sigma^\star \to \mathbb{N}$ where $f(w)$ is the sum of the weights over all computations of M on w. The class #FA is defined as the set of all functions $f : \Sigma^\star \to \mathbb{N}$ that are computed by NFAs.

If needed to distinguish these for different automata, we use a corresponding subscript, for example the weights of computations in M_f would be denoted by $\mathbf{w}_f$, etc.

4.2 Definition (Simple NFA)**.** We say an NFA $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ is *simple* if $\mathrm{im\,wt}, \mathrm{im\,in}, \mathrm{im\,out} \subseteq \{0, 1\}$.

The notion of a simple NFA also motivates our use of the term NFA opposed to $\mathbb{N}$-weighted automaton: We simply count the number of accepting paths of M on a word w. This is in line with #P counting the number of accepting paths on a polynomial time non-deterministic Turing machine.

4.3 Lemma (Folklore)**.** *For every NFA M there exists a simple NFA M' computing the same function.*

[6] Note that in definitions by other authors one can find simpler versions of NFAs, in particular unweighted initial and accepting states and unweighted edges while also restricting to a single initial state. We will see soon that working with unweighted NFAs is not a restriction, but additionally restricting the model to have a single initial state is strictly weaker, since this model could not compute any function f with $f(\varepsilon) > 1$. To obtain the same expressiveness one would have to additionally allow for ε-transitions, while disallowing cycles of ε-transitions to prevent infinite values for f.

[7] We will use both of these terms interchangeably. For a weighted automaton, calling this weight is more natural, while when looking at the underlying graph as a multigraph, multiplicity of paths and walks is more natural.

Proof. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ be an NFA computing f. We construct $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ with

$$Q = Q_f \times [\max(\operatorname{im} \mathrm{wt}_f \cup \operatorname{im} \mathrm{in}_f)] \times [\max(\operatorname{im} \mathrm{out}_f)]$$

$$\mathrm{wt}((q,\alpha,\beta),\sigma,(q',\alpha',\beta')) = \begin{cases} 1 & \text{if } \alpha' \leq \mathrm{wt}_f(q,\sigma,q') \text{ and } \beta = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{in}((q,\alpha,\beta)) = \begin{cases} 1 & \text{if } \alpha \leq \mathrm{in}_f(q) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{out}((q,\alpha,\beta)) = \begin{cases} 1 & \text{if } \beta \leq \mathrm{out}_f(q) \\ 0 & \text{otherwise} \end{cases}$$

Intuitively M consists of multiple identical copies of M_f, where all the weights are handled explicitly as separate computations.

Fix any word $w = w_1 \dots w_n \in \Sigma^\star$. Let $P = q_0 q_1 \dots q_n$ be a computation of M_f on w. Then for $\alpha_0 \in [\mathrm{in}_f(q_0)]$ and $\alpha_i \in [\mathrm{wt}(q_{i-1}, w_i, q_i)]$ for $i \in [n]$ and $\beta_0 = \beta_1 = \ldots = \beta_{n-1} = 1$ and $\beta_n \in [\mathrm{out}_f(q_n)]$ we have that $P' := (q_0,\alpha_0,\beta_0)(q_1,\alpha_1,\beta_1)\dots(q_n,\alpha_n,\beta_n)$ is a computation of M on w of weight exactly 1 while any other choices of the α_i and β_i would lead to a computation of weight 0. In particular there are $\mathrm{in}_f(q_0) \cdot \prod_{i=1}^{n} \mathrm{wt}_f(q_{i-1}, w_i, q_i) \cdot \mathrm{out}_f(q_0)$ non-zero computations in M corresponding to P, which coincides with the weight of P in M_f. Additionally every computation on M can be projected onto a computation on M_f via $(q,i,j) \mapsto q$ and thus M computes f. □

We denote by $a \equiv_p b$ that $a \in \mathbb{N}$ and $b \in \mathbb{N}$ are congruent modulo $p \in \mathbb{N} \setminus \{0\}$. The indicator function $\mathbb{1}_{n=c} : \mathbb{N} \to \mathbb{N}$ is defined as $n \mapsto \begin{cases} 1 & \text{if } n = c \\ 0 & \text{otherwise} \end{cases}$ and analogously for different conditions. By abuse of notation, if we have a function $f : \Sigma^\star \to \mathbb{N}$ and an expression $\varphi : \mathbb{N} \to \mathbb{N}$ in n, we replace n by f in the expression to denote $\varphi \circ f$. For example we use $\mathbb{1}_{f=c}$ to denote the function $w \mapsto \mathbb{1}_{f(w)=c}$, similarly $\binom{f}{2}$ denotes the function $w \mapsto \binom{f(w)}{2}$.

4.3 Functional Closure Properties

4.3.1 Univariate Functional Closure Properties

We say a function $\varphi : \mathbb{N} \to \mathbb{N}$ is a functional closure property of #FA if $\varphi(\#\mathsf{FA}) \subseteq \#\mathsf{FA}$, i.e. if for every function $f \in \#\mathsf{FA}$ the function $\varphi \circ f$ is also in #FA. Our goal in this section is to classify all functional closure properties of #FA. They will be precisely the ultimately PORC functions.

We call a function $\varphi : \mathbb{N} \to \mathbb{N}$ an *ultimately almost PORC function* if there is a *quasiperiod* p, an *offset* $N \in \mathbb{N}$ and *constituents* $\varphi_0, \ldots, \varphi_{p-1} : \mathbb{N} \to \mathbb{Q}$, where each φ_i is either a polynomial with rational coefficients or a function in $2^{\Theta(n)}$, and for every $n \geq N$ we have $\varphi(n) = \varphi_{n \operatorname{rem} p}(n)$. If all the constituents are polynomials, we call φ an *ultimately PORC function*. The smallest representative of each constituent and of the finite cases before the periodic behavior is captured by the *shifted remainder* operator $n \operatorname{srem}_N p$, defined via

$$n \operatorname{srem}_N p = \begin{cases} n & \text{if } n < N \\ \min\{k \geq N \mid k \equiv_p n\} & \text{if } n \geq N \end{cases}$$

The first half of the section is dedicated to proving that every ultimately PORC function is a functional closure property of #FA, see Lemma 4.21. In order to prove this we show that #FA is closed under

4.4 Lemma (Addition)**.** *If $f, g \in \#\mathsf{FA}$, then $f + g \in \#\mathsf{FA}$.*

4.5 Lemma (Multiplication)**.** *If $f, g \in \#\mathsf{FA}$, then $f \cdot g \in \#\mathsf{FA}$.*

4.12 Lemma (Subtraction of constants)**.** *If $f \in \#\mathsf{FA}$, then $\max(f - c, 0) \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

4.13 Lemma (Clamping)**.** *If $f \in \#\mathsf{FA}$, then $\min(f, c) \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

4.14 Lemma (Comparison with constants)**.** *If $f \in \#\mathsf{FA}$, then the functions $\mathbb{1}_{f=c}$, $\mathbb{1}_{f \leq c}$, $\mathbb{1}_{f \geq c}$ are in $\#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

4.16 Lemma (Division by constants)**.** *If $f \in \#\mathsf{FA}$, then $\lfloor f/c \rfloor \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N} \setminus \{0\}$.*

4.17 Lemma (Modular arithmetic)**.** *If $f \in \#\mathsf{FA}$, then the function $\mathbb{1}_{f \equiv_c d}$ is in $\#\mathsf{FA}$ for any constants $c \in \mathbb{N} \setminus \{0\}$ and $d \in \mathbb{Z}_c$.*

4.18 Lemma (Binomial coefficients)**.** *If $f \in \#\mathsf{FA}$, then $\binom{f}{c} \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

While addition and multiplication are technically bivariate functional closure properties, we list them here already since they are abundantly used throughout the proofs of the univariate functional closure properties. Proofs of those two classical results can be found in [DKV09, Ch. 4.1 and 4.2.2], but for the sake of completeness we will also give a proof for them here:

Proof of Lemma 4.4. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ and $M_g = (Q_g, \Sigma, \mathrm{wt}_g, \mathrm{in}_g, \mathrm{out}_g)$ be NFAs computing f and g respectively.

We construct the direct union NFA $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ with

$$\begin{aligned} Q &= Q_f \dot\cup Q_g \\ \mathrm{wt}(q, \sigma, q') &= \begin{cases} \mathrm{wt}_f(q, \sigma, q') & q, q' \in Q_f \\ \mathrm{wt}_g(q, \sigma, q') & q, q' \in Q_g \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{in}(q) &= \begin{cases} \mathrm{in}_f(q) & \text{if } q \in Q_f \\ \mathrm{in}_g(q) & \text{if } q \in Q_g \end{cases} \\ \mathrm{out}(q) &= \begin{cases} \mathrm{out}_f(q) & \text{if } q \in Q_f \\ \mathrm{out}_g(q) & \text{if } q \in Q_g \end{cases} \end{aligned}$$

Any computation with non-zero weight in M is now fully contained in exactly one of the sub-NFA equivalent to M_f or M_g and any computation path on M_f or M_g naturally embeds into M and thus M computes $f + g$. □

Proof of Lemma 4.5. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ and $\mathrm{wt}_g = (Q_g, \Sigma, \mathrm{wt}_g, \mathrm{in}_g, \mathrm{out}_g)$ be NFAs computing f and g respectively. We use the product automaton construction to construct the NFA $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$:

$$\begin{aligned} Q &= Q_f \times Q_g \\ \mathrm{wt}((q_1, q_2), \sigma, (q_1', q_2')) &= \mathrm{wt}_f(q_1, \sigma, q_1') \cdot \mathrm{wt}_g(q_2, \sigma, q_2') \\ \mathrm{in}((q_1, q_2)) &= \mathrm{in}_f(q_1) \cdot \mathrm{in}_g(q_2) \\ \mathrm{out}((q_1, q_2)) &= \mathrm{out}_f(q_1) \cdot \mathrm{out}_g(q_2) \end{aligned}$$

Fix some $w = w_1 \dots w_n \in \Sigma^\star$. Let $\mathcal{P}_f = Q_f^{n+1}$ be all computations of M_f on w and let $\mathcal{P}_g = Q_g^{n+1}$ be all computations of M_g on w. Then $\mathcal{P}_f \times \mathcal{P}_g \cong (Q_f \times Q_g)^{n+1}$ are precisely the computations of M on x by identifying $((q_{f,0} \dots q_{f,n}), (q_{g,0} \dots q_{g,n}))$ with the computation $(q_{f,0}, q_{g,0}) \dots (q_{f,n}, q_{g,n})$. Additionally the computation $(P_f, P_g) \in \mathcal{P}_f \times \mathcal{P}_g$ has weight $\mathbf{w}((P_f, P_g)) = \mathbf{w}_f(P_f) \cdot \mathbf{w}_g(P_g)$ where $\mathbf{w}_f$ and $\mathbf{w}_g$ denote the weights of computations of M_f and M_g respectively. Thus we have

$$\begin{aligned} \Big(\sum_{P_f \in \mathcal{P}_f} \mathbf{w}_f(P_f) \Big) \cdot \Big(\sum_{P_g \in \mathcal{P}_g} \mathbf{w}_g(P_g) \Big) &= \sum_{P_f \in \mathcal{P}_f} \sum_{P_g \in \mathcal{P}_g} \mathbf{w}_f(P_f) \cdot \mathbf{w}_g(P_g) \\ &= \sum_{(P_f, P_g) \in \mathcal{P}_f \times \mathcal{P}_g} \mathbf{w}_f((P_f, P_g)) \end{aligned}$$

and M computes $f \cdot g$. □

With the exception of binomial coefficients all the other closure properties need to be able to "remove" some of the possible computations. For example, consider decrementation, the special case of truncated subtraction by one, and consider some simple NFA M computing some strictly positive function f. We now want to construct an NFA M' that computes $f - 1$, i.e. an NFA that has exactly one non-zero computation less than M (assuming computations of weights zero or one). For this we want a procedure to single out one non-zero computation of M to then change its weight to zero. For stronger models of computation – like polynomial time non-deterministic Turing machines – this approach seems hopeless. Already deciding the existence of one such computation is NP-hard. However for NFAs, deciding the existence of a non-zero computation can be decided by a deterministic finite automaton, namely the powerset automaton. Adjusting the powerset construction to filter out a single non-zero computation, namely the lexicographically minimal one can then be used to show that decrementation is a closure property of #FA.

Generalizing this approach to more general properties about the computations gives us the framework of stepwise computation properties:

4.6 Definition (Stepwise computation property)**.** Let $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be an NFA. A *stepwise computation property* prop is defined as $\mathrm{prop} = (S, \mathrm{init}, \mathrm{step}, \mathrm{cond})$ where S is a finite set and $\mathrm{init} : Q \to S$, $\mathrm{step} : Q \times \Sigma \times Q \times S \to S$ and $\mathrm{cond} : S \to \{0, 1\}$ are functions. For $w = w_1 \dots w_n \in \Sigma^\star$ and a computation $P = q_0 \dots q_n$ of M on w we define a *step sequence* $s_0 := \mathrm{init}(q_0)$ and $s_i := \mathrm{step}(q_{i-1}, w_i, q_i, s_{i-1})$ for $i \in [n]$. We also write $\mathrm{prop}(w, P) := \mathrm{cond}(s_n)$ to be the evaluation of the property.

These stepwise computation properties now enable us, given a simple NFA, to construct NFAs computing both of the following:

4.7 Lemma. *Let M_f be a simple NFA computing a function f and let* prop *be a stepwise computation property. Then there is an NFA M computing $g(w) = \sum_P \mathbf{w}_f(P) \cdot \mathrm{prop}(w, P)$, where the sum is over all computations P of M_f on w.*

Proof. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$. We construct the NFA $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ with $Q = Q_f \times S$ and

$$\mathrm{in}((q, s)) = \begin{cases} \mathrm{in}_f(q) & \text{if } s = \mathrm{init}(q) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{out}((q, s)) = \mathrm{out}_f(q) \cdot \mathrm{cond}(s)$$

$$\mathrm{wt}((q, s), \sigma, (q', s')) = \begin{cases} \mathrm{wt}_f(q, \sigma, q') & \text{if } s' = \mathrm{step}(q, \sigma, q', s) \\ 0 & \text{otherwise} \end{cases}$$

Fix some $w = w_1 \dots w_n \in \Sigma^\star$. Any computation $P = (q_0, s_0) \dots (q_n, s_n)$ of M on w directly corresponds to the computation $q_0 \dots q_n$ of M_f. If $s_0 \neq \mathrm{init}(q_0)$ or if $s_i \neq \mathrm{step}(q_{i-1}, w_i, q_i, s_{i-1})$ for any $i \in [n]$ then P has weight 0. Otherwise P has weight

$\mathbf{w}(P) = \mathbf{w}_f(q_0 \dots q_n) \cdot \text{cond}(s_n) = \mathbf{w}_f(q_0 \dots q_n) \cdot \text{prop}(w, q_0 \dots q_n)$. The claim on the function computed by M directly follows. □

4.8 Lemma. *Let M_f be a simple NFA computing a function f and let* prop *be a stepwise computation property. Then there is an NFA M computing $g(w) = \sum_P \text{prop}(w, P)$, where the sum is over all computations P of M_f on w.*

Proof. Let $M_f = (Q_f, \Sigma, \text{wt}_f, \text{in}_f, \text{out}_f)$. We construct the NFA $M = (Q, \Sigma, \text{wt}, \text{in}, \text{out})$ with $Q = Q_f \times S$ and

$$\text{in}((q,s)) = \begin{cases} 1 & \text{if } s = \text{init}(q) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{out}((q,s)) = \text{cond}(s)$$

$$\text{wt}((q,s),\sigma,(q',s')) = \begin{cases} 1 & \text{if } s' = \text{step}(q,\sigma,q',s) \\ 0 & \text{otherwise} \end{cases}$$

Any computation $P = (q_0, s_0) \dots (q_n, s_n)$ of M directly corresponds to the computation $q_0 \dots q_n$ of M_f. If $s_0 \neq \text{init}(q_0)$ or if $s_i \neq \text{step}(q_{i-1}, w_i, q_i, s_{i-1})$ for any $i \in [n]$ then P has weight 0. Otherwise P has weight $\mathbf{w}(P) = \text{cond}(s_n) = \text{prop}(w, q_0 \dots q_n)$. The claim on the function computed by M directly follows. □

In other words, stepwise computation properties allow us to either "disable" specific computations of M_f or they allow us to directly extract information about the computations of M_f. Note that the restriction on the finiteness of S is necessary, as the elements of S are hard-coded into the state space of the NFA in Lemmas 4.7 and 4.8. In particular, these lemmas do not hold for even countably infinite S, which can be seen with the example $S = \mathbb{N}$ when we define the stepwise computation property in such a way that $\text{prop}(w, P) = 1$ iff $|w|$ is prime, leading to an NFA recognizing the language of all words of prime length, a well known contradiction.

Further note that these two constructions do not incur exponential blowups themselves, however for most of our applications the set S will be of exponential size in the number of states of M_f.

Returning to our decrementation example, to use Lemma 4.7 we want to construct a stepwise computation property prop with $\text{prop}(w, P) = 0$ iff P is the lexicographically smallest non-zero weight computation on w. For this we can set $S = \mathcal{P}(Q)$ to be the set of all subsets of states Q, denoting the set of states that currently are the endpoints of lexicographically smaller partial computations of non-zero weight than the partial computation P we are on. We need to store all such potential states, since some current partial non-zero weight computations might not be possible to be completed to a full non-zero weight computation. Initially this set contains all states that are smaller than

the start state of P, the step function then checks which lexicographically smaller partial computations can be extended and whether any new partial computations that agreed with P up to this state can be lexicographically smaller than P. Finally the cond function then checks whether there are any such lexicographically smaller partial computations left that can be completed to a non-zero weight computation, i.e. that end on a state $q \in Q$ with $\mathrm{out}(q) = 1$.

We want to generalize this idea to be able to generally create stepwise computation properties that argue about the number of non-zero computations, either in total or lexicographically smaller than a given computation. However doing this in general would require choosing the set S as the set of functions $Q \to \mathbb{N}$, which is infinite. As a result we embed the number of computations into finite semirings first to then extract the relevant information. For this purpose we only consider semirings with both additive and multiplicative identities. Homomorphisms h from a semiring $\mathcal{R}$ into another semiring $\mathcal{R}'$ need to fulfill $h(a+b) = h(a)+h(b)$, $h(a\cdot b) = h(a)\cdot h(b)$, $h(1_{\mathcal{R}}) = 1_{\mathcal{R}'}$, $h(0_{\mathcal{R}}) = 0_{\mathcal{R}'}$. Since every element of $\mathbb{N}$ is either 0 or can be formed by repeated addition of 1, any homomorphism from $\mathbb{N}$ into any other semiring is uniquely defined.

We can then use $\mathcal{R}$ to construct stepwise computation properties. Together with Lemmas 4.7 and 4.8, these use similar ideas to [KLMP04].

4.9 Lemma. *Let $N = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be a simple NFA and let $\mathcal{R}$ be a finite semiring and let $\tau : \mathbb{N} \to \mathcal{R}$ be the unique homomorphism from $\mathbb{N}$ to $\mathcal{R}$. For any function $\pi : \mathcal{R} \to \{0, 1\}$ there is a stepwise computation property* prop *with* $\mathrm{prop}(w, P) = \pi(\tau(\sum_{P'} \mathbf{w}(P')))$*, where the sum is over all computations P' of N on w, independent of P.*

Proof. We construct $\mathrm{prop} = (S, \mathrm{init}, \mathrm{step}, \mathrm{cond})$ as follows:

$$\begin{aligned} S &= Q \to \mathcal{R} \\ \mathrm{init}(q) &= r \mapsto \tau(\mathrm{in}(r)) \\ \mathrm{step}(q, \sigma, q', s) &= r \mapsto \textstyle\sum_{r' \in Q} s(r') \cdot \tau(\mathrm{wt}(r', \sigma, r)) \\ \mathrm{cond}(s) &= \pi\Big(\textstyle\sum_{r \in Q} s(r) \cdot \tau(\mathrm{out}(r))\Big) \end{aligned}$$

Note that step and init depend on neither q nor q'. So for a fixed $w = w_1 \dots w_n \in \Sigma^\star$ the step sequence $s_0, \dots, s_n$ collapses to $s_0 = r \mapsto \tau(\mathrm{in}(r))$ and $s_i = r \mapsto \sum_{r' \in Q} s_{i-1}(r') \cdot \tau(\mathrm{wt}(r', w_i, r))$ for $i \in [n]$, independent of the specific computation of N on x.

We first prove that for all $w = w_1 \dots w_n \in \Sigma^\star$ and $q \in Q$ we have $s_n(r) = \tau(\sum_{P_r} \underline{\mathbf{w}}(P_r))$ where the sum is over all computations $P_r = q_0 \dots q_n$ of N on w with $q_n = r$. We prove

this by induction over n. For $n = 0$ this holds directly. For $n > 0$ we have

$$\begin{aligned} s_n(r) &= \sum_{q_{n-1} \in Q} s_{n-1}(q_{n-1}) \cdot \tau(\mathrm{wt}(q_{n-1}, w_n, r)) \\ &= \sum_{q_{n-1} \in Q} \tau\Big(\sum_{q_0,\dots,q_{n-2} \in Q} \underline{\mathbf{w}}(q_0 \dots q_{n-1}) \Big) \cdot \tau(\mathrm{wt}(q_{n-1}, w_n, r)) \\ &= \tau\Big(\sum_{q_{n-1} \in Q} \sum_{q_0,\dots,q_{n-2} \in Q} \underline{\mathbf{w}}(q_0 \dots q_{n-1}) \cdot \mathrm{wt}(q_{n-1}, w_n, r) \Big) \\ &= \tau\Big(\sum_{q_0,\dots,q_{n-1} \in Q} \underline{\mathbf{w}}(q_0 \dots q_{n-1} r) \Big) . \end{aligned}$$

We can thus finish the proof with

$$\begin{aligned} \mathrm{prop}(w, P) &= \mathrm{cond}(s_n) \\ &= \pi\Big(\sum_{q_n \in Q} s_n(r) \cdot \tau(\mathrm{out}(q_n)) \Big) \\ &= \pi\Big(\sum_{q_n \in Q} \tau\Big(\sum_{q_0,\dots,q_{n-1} \in Q} \underline{\mathbf{w}}(q_0 \dots q_n) \Big) \cdot \tau(\mathrm{out}(q_n)) \Big) \\ &= \pi\Big(\tau\Big(\sum_{q_n \in Q} \sum_{q_0,\dots,q_{n-1} \in Q} \underline{\mathbf{w}}(q_0 \dots q_n) \cdot \mathrm{out}(q_n) \Big) \Big) \\ &= \pi\Big(\tau\Big(\sum_{q_0,\dots,q_n \in Q} \mathbf{w}(q_0 \dots q_n) \Big) \Big) . \end{aligned}$$

□

4.10 Lemma. *Let $N = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be a simple NFA with some ordering $<$ of Q, let $\mathcal{R}$ be a finite semiring and let $\tau : \mathbb{N} \to \mathcal{R}$ be the unique homomorphism from $\mathbb{N}$ to $\mathcal{R}$. For any function $\pi : \mathcal{R} \to \{0, 1\}$ there is a stepwise computation property* prop *with* $\mathrm{prop}(w, P) = \pi(\tau(\sum_{P'} \mathbf{w}(P')))$ *for all non-zero computations P of N on w, where the sum is over all computations P' of N on w that are lexicographically smaller than P.*

Proof. We construct $\mathrm{prop} = (S, \mathrm{init}, \mathrm{step}, \mathrm{cond})$ as follows:

$$\begin{aligned} S &= Q \to \mathcal{R} \\ \mathrm{init}(q) &= r \mapsto \tau(\mathbb{1}_{r<q} \cdot \mathrm{in}(r)) \\ \mathrm{step}(q, \sigma, q', s) &= r \mapsto \textstyle\sum_{r' \in Q} s(r') \cdot \tau(\mathrm{wt}(r', \sigma, r)) + \tau(\mathbb{1}_{r<q'} \cdot \mathrm{wt}(q, \sigma, r)) \\ \mathrm{cond}(s) &= \pi\left(\textstyle\sum_{r \in Q} s(r) \cdot \tau(\mathrm{out}(r)) \right) \end{aligned}$$

Fix some $w = w_1 \dots w_n \in \Sigma^\star$ and some computation $P = q_0 \dots q_n$ of N on w of non-zero weight. This fixes the step sequence $s_0, \dots, s_n$. We now prove by induction over i that $s_i(r) = \sum_{P'} \tau(\underline{\mathbf{w}}(P'))$ for all $i \in \{0, \dots, n\}$ and $r \in Q$, where the sum is over all computations $P' = q'_0 \dots q'_i$ of $w_1 \dots w_i$ with $q'_i = r$ that are lexicographically smaller than $q_0 \dots q_i$. For $i = 0$ this is trivially the case since the only computations $P' = q'_0$ that are lexicographically smaller than the computation q_0 are the ones with $q'_0 < q_0$. For $i > 0$ for a computation $P' = q'_0 \dots q'_i$ to be lexicographically smaller than $q_0 \dots q_i$ there are two possibilities. Either $q'_0 \dots q'_{i-1}$ is already lexicographically smaller than $q_0 \dots q_{i-1}$ or $q'_0 \dots q'_{i-1} = q_0 \dots q_{i-1}$ and $q'_i < q_i$. In the second case the weight of P' is precisely $\mathrm{wt}(q_{i-1}, w_i, q'_i)$ since P is a computation of non-zero weight and thus weight exactly 1. Combining all of this we can finish the proof of the claim with a similar argument to Lemma 4.9:

$$
\begin{aligned}
s_i(r) &= \sum_{q'_{i-1} \in Q} s_{i-1}(q'_{i-1}) \cdot \tau(\mathrm{wt}(q'_{i-1}, w_i, r)) + \tau(\mathbb{1}_{r<q_i} \cdot \mathrm{wt}(q_{i-1}, \sigma, r)) \\
&= \tau\Big(\sum_{\substack{q'_0 \dots q'_{i-1} \\ \text{lex. smaller than} \\ q_0 \dots q_{i-1}}} \underline{\mathbf{w}}(q'_0 \dots q'_{i-1})) \cdot \mathrm{wt}(q'_{i-1}, w_i, r) \Big) + \tau(\mathbb{1}_{r<q_i} \cdot \underline{\mathbf{w}}(q_0 \dots q_{i-1} r)) \\
&= \tau\Big(\sum_{\substack{q'_0 \dots q'_{i-1} \\ \text{lex. smaller than} \\ q_0 \dots q_{i-1}}} \underline{\mathbf{w}}(q'_0 \dots q'_{i-1} r) + \mathbb{1}_{r<q_i} \cdot \underline{\mathbf{w}}(q_0 \dots q_{i-1} r)) \Big) \\
&= \tau\Big(\sum_{\substack{q'_0 \dots q'_{i-1} r \\ \text{lex. smaller than} \\ q_0 \dots q_{i-1} q_i}} \underline{\mathbf{w}}(q'_0 \dots q'_{i-1} r) \Big)
\end{aligned}
$$

The statement of this lemma then follows analogously to Lemma 4.9. □

Note that the previous lemma makes no statement about the value of $\mathrm{prop}(w, P)$ for any computations P of weight zero. However, this is enough for our uses, since we only combine it with Lemma 4.7, i.e., $\mathrm{prop}(w, P)$ gets weighted by $\mathbf{w}(P)$.

Most commonly, as is the case for decrementation, we want to be able to exactly distinguish the number of non-zero computations if it is less than k and otherwise be able to tell that the number is at least k. This is achieved by using the following capped semiring:

4.11 Definition (Capped semiring)**.** For $k \in \mathbb{N}$ we call the semiring $\mathcal{R}_k = \{0, \dots, k\}$ with the operations $a +_{\mathcal{R}} b := \min(a + b, k)$ and $a \cdot_{\mathcal{R}} b := \min(a \cdot b, k)$ the *capped* semiring.

We can now show that decrementation is a functional closure property by simply using Lemma 4.10 using the capped semiring $\mathcal{R}_1$ and $\pi(a) = a$ to construct our wanted stepwise computation property computing $\mathrm{prop}(w, P) = 0$ iff P is the lexicographically smallest non-zero weight computation on w.

Most of the remaining closure properties are now proven by using the capped semiring of a specific size and choosing the function π accordingly.

4.12 Lemma (Subtraction of constants)**.** *If $f \in \#\mathsf{FA}$, then $\max(f - c, 0) \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

Proof. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ be a simple NFA computing f with an arbitrary ordering $<$ on Q_f. Lemma 4.10 on the capped semiring $\mathcal{R}_c$ with $\pi(a) = \mathbb{1}_{a \geq c}$ for all $a \in \mathcal{R}$ constructs a stepwise computation property prop with

$$\mathrm{prop}(w, P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ that are lex.} \\ & \text{smaller than } P \text{ is at least } c \\ 0 & \text{otherwise} \end{cases}$$

for all computations P on w of non-zero weight, i.e., $\mathrm{prop}(w, P) = 0$ iff P is one of the c lexicographically smallest computations on w with non-zero weight. It follows that the NFA M constructed by Lemma 4.7 computes $g(w) = \max(f(w) - c, 0)$. □

If instead of rejecting the c lexicographically smallest computations, we accept only those computations, we compute the minimum of f and c.

4.13 Lemma (Clamping)**.** *If $f \in \#\mathsf{FA}$, then $\min(f, c) \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

Proof. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ be a simple NFA computing f with an arbitrary ordering $<$ on Q_f. Lemma 4.10 on the capped semiring $\mathcal{R}_c$ with $\pi(a) = \mathbb{1}_{a < c}$ for all $a \in \mathcal{R}$ constructs a stepwise computation property prop with

$$\mathrm{prop}(w, P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ that are lex.} \\ & \text{smaller than } P \text{ is at most } c - 1 \\ 0 & \text{otherwise} \end{cases}$$

for all computations P of M_f on w of non-zero weight, i.e. $\mathrm{prop}(w, P) = 1$ iff P is one of the c lexicographically smallest computations on w with non-zero weight. It follows that the NFA M constructed by Lemma 4.7 computes $g(w) = \min(f(w), c)$. □

By using the capped semiring $\mathcal{R}_{c+1}$ with $\pi_=(a) = \mathbb{1}_{a=c}$, $\pi_\leq(a) = \mathbb{1}_{a \leq c}$ and $\pi_\geq(a) = \mathbb{1}_{a \geq c}$, we can compute the indicator functions $\mathbb{1}_{f=c}$, $\mathbb{1}_{f \leq c}$ and $\mathbb{1}_{f \geq c}$ respectively.

4.14 Lemma (Comparison with constants)**.** *If $f \in \#\mathsf{FA}$, then the functions $\mathbb{1}_{f=c}$, $\mathbb{1}_{f \leq c}$, $\mathbb{1}_{f \geq c}$ are in $\#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

Proof. Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ be a simple NFA computing f with an arbitrary ordering $<$ on Q_f. Lemma 4.9 on the capped semiring $\mathcal{R}_{c+1}$ with $\pi_=(a) = \mathbb{1}_{a=c}$, $\pi_\leq(a) = \mathbb{1}_{a\leq c}$ and $\pi_\geq(a) = \mathbb{1}_{a\geq c}$ for all $a \in \mathcal{R}$ constructs stepwise computation properties $\mathrm{prop}_=$, $\mathrm{prop}_\leq$ and $\mathrm{prop}_\geq$ respectively with

$$\mathrm{prop}_=(w,P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ is exactly } c \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{prop}_\leq(w,P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ is at most } c \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{prop}_\geq(w,P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ is at least } c \\ 0 & \text{otherwise} \end{cases}$$

for all computations P of M_f on w. It follows that the NFAs $M_=, M_\leq$ and $M_\geq$ constructed by Lemma 4.8 compute $g_=(w) = \mathbb{1}_{f(w)=c} \cdot |Q_f|^{|w|+1}$, $g_\leq(w) = \mathbb{1}_{f(w)\leq c} \cdot |Q_f|^{|w|+1}$ and $g_\geq(w) = \mathbb{1}_{f(w)\geq c} \cdot |Q_f|^{|w|+1}$ respectively. Since $|Q_f|^{|w|+1}$ is always positive Lemma 4.13 finishes the proof with

$$\min(g_=(w),1) = \mathbb{1}_{f(w)=c} \quad \min(g_\leq(w),1) = \mathbb{1}_{f(w)\leq c} \quad \min(g_\geq(w),1) = \mathbb{1}_{f(w)\geq c}\,. \qquad \square$$

The previous lemma in particular also implies the following:

4.15 Lemma. *If $\varphi : \mathbb{N} \to \mathbb{N}$ is a functional closure property of #FA and $\psi : \mathbb{N} \to \mathbb{N}$ is an arbitrary function with $\varphi(n) = \psi(n)$ for all but finitely many $n \in \mathbb{N}$, then ψ is also a functional closure property of #FA.*

Proof. Let $f \in \#\mathsf{FA}$ be arbitrary. Further let $N \in \mathbb{N}$ be such that $\varphi(n) = \psi(n)$ for all $n \geq N$. Then $(\psi \circ f)(w) = \sum_{i=0}^{N-1} \mathbb{1}_{f(w)=i}\psi(i) + \mathbb{1}_{f(w)\geq N}\varphi(f(w))$ for all $w \in \Sigma^\star$. In particular $\psi \circ f \in \#\mathsf{FA}$ since the $\psi(i)$ are constants and $\varphi \circ f \in \#\mathsf{FA}$. $\square$

Division and modular arithmetic however use a different semiring: they use the finite cyclic semiring $\mathbb{Z}_c = \{0, \ldots, c-1\}$ with $\pi(a) = \mathbb{1}_{a=c-1}$ and $\pi(a) = \mathbb{1}_{a=d}$ respectively.

4.16 Lemma (Division by constants)**.** *If $f \in \#\mathsf{FA}$, then $\lfloor f/c \rfloor \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N} \setminus \{0\}$.*

Proof. For $c = 1$ the statement is trivially true, so let $c \geq 2$.
Let $M_f = (Q_f, \Sigma, \mathrm{wt}_f, \mathrm{in}_f, \mathrm{out}_f)$ be a simple NFA computing f with an arbitrary ordering $<$ on Q_f. We use the finite cyclic semiring $\mathcal{R} = \mathbb{Z}_c$ with the usual addition and

multiplication. Lemma 4.10 with $\pi(a) = \mathbb{1}_{a=c-1}$ for all $a \in \mathcal{R}$ constructs a stepwise computation property prop with

$$\text{prop}(w, P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \text{ that are} \\ & \text{lex. smaller than } P \text{ is one less than a multiple of } c \\ 0 & \text{otherwise} \end{cases}$$

for all computations P of M_f on w of non-zero weight. It follows that the NFA M constructed by Lemma 4.7 computes $g(w) = \left\lfloor \frac{f(w)}{c} \right\rfloor$. □

4.17 Lemma (Modular arithmetic)**.** *If $f \in \#\mathsf{FA}$, then the function $\mathbb{1}_{f \equiv_c d}$ is in $\#\mathsf{FA}$ for any constants $c \in \mathbb{N} \setminus \{0\}$ and $d \in \mathbb{Z}_c$.*

Proof. For $c = 1$ the statement is trivially true, so let $c \geq 2$.
Let $M_f = (Q_f, \Sigma, \text{wt}_f, \text{in}_f, \text{out}_f)$ be a simple NFA computing f with an arbitrary ordering $<$ on Q_f. We use the finite cyclic semiring $\mathcal{R} = \mathbb{Z}_c$ with the usual addition and multiplication. Lemma 4.9 with $\pi(a) = \mathbb{1}_{a=d}$ for all $a \in \mathcal{R}$ constructs a stepwise computation property prop with

$$\text{prop}(w, P) = \begin{cases} 1 & \text{if the number of non-zero computations } P' \text{ on } w \\ & \text{is } d \text{ more than a multiple of } c \\ 0 & \text{otherwise} \end{cases}$$

for all computations P of M_f on w of non-zero weight. It follows that the NFA M constructed by Lemma 4.8 computes $g(w) = \mathbb{1}_{f(w) \equiv_c d} \cdot |Q_f|^{|w|+1}$. Since $|Q_f|^{|w|+1}$ is always positive Lemma 4.13 finishes the proof with $\min(g(w), 1) = \mathbb{1}_{f(w) \equiv_c d}$. □

The previous closure properties turn out to already be sufficient to generate all functional closure properties, so in particular they are sufficient to generate binomial coefficients by using subtraction of constants, multiplication and division by constants by using the definition of binomial coefficients as a polynomial: $\binom{x}{c} = \frac{1}{c!} x \cdot (x-1) \cdot \ldots \cdot (x-c+1)$. Nonetheless, we give an additional proof for binomial coefficients as a different interesting application of the stepwise computation property framework.

4.18 Lemma (Binomial coefficients)**.** *If $f \in \#\mathsf{FA}$, then $\binom{f}{c} \in \#\mathsf{FA}$ for any constant $c \in \mathbb{N}$.*

Proof. Let $M_f = (Q_f, \Sigma, \text{wt}_f, \text{in}_f, \text{out}_f)$ be a simple NFA computing f and let $c \in \mathbb{N}$. For $c < 2$ the statement of this lemma is trivially true, so assume $c \geq 2$.

We construct the c-fold product automaton $M_f^c = (Q_f^c, \Sigma, \text{wt}_f^c, \text{in}_f^c, \text{out}_f^c)$ with

$$\begin{aligned}\mathrm{wt}_f^c((q_1,\dots,q_c),\sigma,(q'_1,\dots,q'_c)) &= \textstyle\prod_{i=1}^c \mathrm{wt}_f(q_i,\sigma,q'_i)\\ \mathrm{in}_f^c((q_1,\dots,q_c)) &= \textstyle\prod_{i=1}^c \mathrm{in}_f(q_i)\\ \mathrm{out}_f^c((q_1,\dots,q_c)) &= \textstyle\prod_{i=1}^c \mathrm{out}_f(q_i)\end{aligned}$$

M_f^c is a simple NFA and every computation on M_f^c is the cartesian product of c computations on M_f. Our aim is to now construct a stepwise computation property $\mathrm{prop} = (S, \mathrm{init}, \mathrm{step}, \mathrm{cond})$ such that $\mathrm{prop}(w, P) = 1$ iff P is composed of c pairwise distinct computations[8] on N_f.

[8] We could also require them to be sorted in lexicographical order by having S be the set of all total preorders, but since we can divide by $c!$ we are going with the easier exposition.

For this let S be the set of all equivalence relations on the set $[c]$. We then define $\mathrm{init}((q_0,\dots,q_c))$ to be the equivalence relation R_0 with $(a,b) \in R_0$ iff $q_a = q_b$. Additionally we define $\mathrm{cond}(R) = 1$ iff R is the equivalence relation where every element is only equivalent to itself, i.e. a computation gets accepted iff all its constituent computations are pairwise distinct. Finally we define $\mathrm{step}((q_1,\dots,q_c),\sigma,(q'_1,\dots,q'_c),R)$ to be the equivalence relation R' defined via $(a,b) \in R'$ iff $(a,b) \in R$ and $q'_a = q'_b$. With a simple induction we can prove that for a computation $P = P_1 \times \ldots \times P_c$ and the step sequence $R_0,\dots,R_n$ we have $(a,b) \in R_i$ iff the computations P_a and P_b are identical for the first i steps. It follows that the NFA M constructed by Lemma 4.7 computes $g(w) = \binom{f(w)}{c} \cdot c!$. Now, $\binom{f}{c} \in \#\mathsf{FA}$ by Lemma 4.16. □

While the combination of the previous lemmas can be used to show that any polynomial written in the binomial basis with non-negative integer coefficients is a functional closure property of #FA, we can do better by considering a shifted binomial basis. For example, consider the polynomial $\varphi(x) = \frac{x^2}{2} - \frac{3x}{2} + 1$. This polynomial is non-negative for all $x \in \mathbb{N}$. Writing φ in the binomial basis we get $\varphi(x) = \binom{x}{2} - \binom{x}{1} + 1$. If however we allow the upper indices of the binomial basis to be shifted, we can write φ without the use of negative coefficients as $\varphi(x) = \binom{x-1}{2}$. While $x-1$ itself is not a functional closure property of #FA the function $\max(x-1,0)$ is a functional closure property of #FA and is different from $x-1$ for only finitely many $x \in \mathbb{N}$. In the same way, we see that $\varphi'(x) := \binom{\max(x-1,0)}{2}$ only differs from φ for finitely many $x \in \mathbb{N}$, namely $x = 0$. Using Lemma 4.15 to change those finitely many values, we see that φ is indeed a functional closure property of #FA.

Generalizing this idea we will show with the next two lemmas that this is possible for any φ with integer coefficients in the binomial basis, with a small restriction: We don't show that φ itself is a functional closure property of #FA, but rather that $x \mapsto \max(\varphi(x), 0)$ is one. Note that this restriction is the best we can hope for, since no computation in an NFA can ever have negative weight.

4.19 Lemma. *Let $\varphi(x) = \sum_{i=0}^r a_i \cdot \binom{x}{i}$ with $a_i \in \mathbb{Z}$ and $a_r > 0$. Then there are $b_0,\dots,b_r \in \mathbb{N}$ and $c_0,\dots,c_r \in \mathbb{N}$ with $\varphi(x) = \sum_{i=0}^r b_i \cdot \binom{x-c_i}{i}$.*

Proof. We prove this via induction on r. For $r = 0$ we can choose $b_0 = a_0$ and $c_0 = 0$ to see that φ is of the required form. For $r > 0$ we choose $b_r = a_r$ and $c_r = \max\{-a_{r-1} + 1, 0\}$.

We define the polynomial $\varphi'(x) := \varphi(x) - b_r \cdot \binom{x-c_r}{r}$ and analyze the leading binomial $\binom{x-c_r}{r}$. Using the Chu-Vandermonde identity [Spi16] we see $\binom{x-c_r}{r} = \sum_{i=0}^{r} \binom{-c_r}{r-i}\binom{x}{i} = \sum_{i=0}^{r}(-1)^{r-i}\binom{r-i+c_r-1}{r-i}\binom{x}{i}$.

Combining this with $\varphi(x) = \sum_{i=0}^{r} a_i \cdot \binom{x}{i}$ we see that $\varphi(x) - b_r \cdot \binom{x-c_r}{r} = \sum_{i=0}^{r-1} a'_i \cdot \binom{x}{i}$ with $a'_i = a_i - b_r \cdot (-1)^{r-i} \cdot \binom{r-i+c_r-1}{r-i}$. Furthermore we have $a'_{r-1} > 0$ due to $b_r \cdot (-1)^{r-(r-1)}\binom{r-(r-1)+c_r-1}{r-(r-1)} = -b_r c_r < a_{r-1}$. By induction there are $b'_0, \ldots, b'_{r-1} \in \mathbb{N}$ and $c'_0, \ldots, c'_{r-1} \in \mathbb{N}$ s.t. $\varphi'(x) = \sum_{i=0}^{r-1} b'_i \cdot \binom{x-c'_i}{i}$. Thus by choosing $c_i = c'_i$ and $b_i = b'_i$ for $i \in \{0, \ldots, r-1\}$ we get $\varphi(x) = \sum_{i=0}^{r} b_i \cdot \binom{x-c_i}{i}$. □

4.20 Lemma (Integer-valued polynomials)**.** *Let $f \in \#\mathsf{FA}$ and let $\varphi : \mathbb{Q} \to \mathbb{Q}$ be an integer-valued polynomial, then $\max(\varphi \circ f, 0) \in \#\mathsf{FA}$.*

Proof. We can assume the leading coefficient of φ to be positive. Otherwise $\max(\varphi \circ f, 0)$ can be directly written as a finite sum $\sum_i c_i \cdot \mathbb{1}_{f=i}$ which is in $\#\mathsf{FA}$ by Lemmas 4.4, 4.5 and 4.14. Write φ in the binomial basis as $\varphi(x) = a_0 \cdot \binom{x}{0} + \ldots + a_r \cdot \binom{x}{r}$ with $a_r > 0$. Since φ is integer-valued, all of the a_i are integers, see [IP22, Prop. 4.2.1]. Using Lemma 4.19 we get a representation $\varphi(x) = \sum_{i=0}^{r} b_i \cdot \binom{x-c_i}{i}$ with $b_0, \ldots, b_r \in \mathbb{N}$ and $c_0, \ldots, c_r \in \mathbb{N}$. For $x \geq \max\{c_i \mid 0 \leq i \leq r\} =: N$ we have that $\binom{x-c_i}{i} = \binom{\max(x-c_i,0)}{i}$ and thus $\psi(x) := \sum_{i=0}^{r} b_i \cdot \binom{\max(x-c_i,0)}{i}$ only differs from $x \mapsto \max(\varphi(x), 0)$ on finitely many inputs and is a functional closure property of $\#\mathsf{FA}$ by Lemmas 4.4, 4.5, 4.12 and 4.18. Lemma 4.15 then finishes off the claim. □

We now have the tools available to show our claim that every ultimately PORC function is a functional closure property of $\#\mathsf{FA}$.

4.21 Lemma. *Every ultimately PORC function is a functional closure property of $\#\mathsf{FA}$.*

Proof. Let $\varphi : \mathbb{N} \to \mathbb{N}$ be an ultimately PORC function with period p comprised of the polynomial constituents $\varphi_0, \ldots, \varphi_{p-1} : \mathbb{N} \to \mathbb{Q}$ and $N \in \mathbb{N}$, s.t. for all $n \geq N$ we have $\varphi(n) = \varphi_{n \operatorname{rem} p}(n)$. Additionally let $f \in \#\mathsf{FA}$. We can write

$$\varphi \circ f = \sum_{i=0}^{N-1} \mathbb{1}_{f=i} \cdot \varphi(i) + \mathbb{1}_{f \geq N} \cdot \left(\sum_{i=0}^{p-1} \mathbb{1}_{f \equiv_p i} \cdot \lfloor \max(\varphi_i \circ f, 0) \rfloor\right).$$

Combining Lemmas 4.4, 4.5, 4.14 and 4.17 this shows $\varphi \circ f \in \#\mathsf{FA}$, if we can show $\lfloor \max(\varphi_i \circ f, 0) \rfloor \in \#\mathsf{FA}$ for all $i \in \{0, \ldots, p-1\}$. To show this let α_i be the common denominator of the coefficients of φ_i. Then $\alpha_i \cdot \varphi_i$ is a polynomial with integer coefficients, so it in particular is an integer-valued polynomial and by Lemma 4.20 we have that

$\max(\alpha_i \cdot \varphi_i \circ f, 0) \in \#\mathsf{FA}$. Combining this with Lemma 4.16 we get that $\left\lfloor \frac{\max(\alpha_i \cdot \varphi_i \circ f, 0)}{\alpha_i} \right\rfloor = \lfloor \max(\varphi_i \circ f, 0) \rfloor \in \#\mathsf{FA}$. □

The remainder of this section is dedicated to showing that no other functional closure properties of #FA exist. This will make use of the following well known algebraic interpretation of NFAs:

4.22 Lemma (see [Sch61])**.** *If $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ is an NFA, then there are matrices $A_\sigma \in \mathbb{N}^{|Q| \times |Q|}$ for each symbol $\sigma \in \Sigma$ and vectors $a, b \in \mathbb{N}^{|Q|}$, s.t. M computes $a^T \cdot \left(\prod_{j=1}^{|w|} A_{w_j}\right) \cdot b$ for all $w \in \Sigma^\star$.*

Proof. We index A_σ, a and b using states $q, q' \in Q$. Choose $(A_\sigma)_{q,q'} = \mathrm{wt}(q, \sigma, q')$, $a_q = \mathrm{in}(q)$ and $b_q = \mathrm{out}(q)$. It is now easy to see that M computes exactly $a^T \cdot \left(\prod_{j=1}^{|w|} A_{w_j}\right) \cdot b$. □

When restricting to a unary alphabet $\Sigma = \{\sigma\}$, this degenerates the computed function to $a^T \cdot A_\sigma^{|w|} \cdot b$. In order to analyze the behavior of these functions we first analyze the behavior of the matrix power as a function in $|w|$ in the next two rather technical lemmas.

4.23 Lemma. *Let $A \in \mathbb{N}^{k \times k}$. Then any diagonal entry of A^n is a function $f : \mathbb{N} \to \mathbb{N}$ with one of the following properties:*

1. *$f(n) = 0$ for all $n \in \mathbb{N} \setminus \{0\}$ and $f(0) = 1$.*
2. *There is a $p \in \mathbb{N} \setminus \{0\}$, such that for all $n \in \mathbb{N}$ we have $f(n) = \mathbb{1}_{n \equiv_p 0}$.*
3. *There is a $p \in \mathbb{N} \setminus \{0\}$ and a function $g \in 2^{\Theta(n)}$, such that for all $n \in \mathbb{N}$ we have $f(n) = \mathbb{1}_{n \equiv_p 0} \cdot g(n)$.*

These naturally correspond to vertices v in the multigraph defined by the adjacency matrix A with

1. no paths from v to v.
2. exactly one path from v to v of length p.
3. multiple walks from v to v where the lengths of all the walks from v to v have gcd p.

Proof. To prove the lemma, let v be arbitrary and let $f_v(n) = (A^n)_{vv}$. Note that we always have $f_v(0) = 1$ since A^0 is the identity matrix. We split into three cases:

$f_v(n) = 0$ **for all** $n > 0$**:** In this case we are done.

$f_v(n) \in \{0,1\}$ **for all** $n \in \mathbb{N}$ **and there is some** $\ell > 0$ **with** $f_v(\ell) = 1$**:** Let $p \geq 1$ be minimal with $f_v(p) = 1$. Then we directly know that for any multiple $i \cdot p$ we have $f_v(ip) = (A^{ip})_{vv} \geq ((A^p)_{vv})^i = f_v(p)^i = 1$. In the setting if A is interpreted as the adjacency matrix of a multigraph, this corresponds to a walk consisting of repeating the same path of length p from v to itself i times.

Assume for the sake of contradiction there is any other n that is not a multiple of p with $f_v(n) = 1$. Then we have that $f_v(n \cdot p) = (A^{np})_{vv} \geq (A^p)^n_{vv} + (A^n)^p_{vv} = f_v(p)^n + f_v(n)^p = 2$, a contradiction. Going back to the graph setting, this corresponds to the choice of either repeating a walk from v to v of length n a total of p times or repeating a walk from v to v of length p a total of n times, leading to two different walks of length $n \cdot p$ from v to v since neither does p divide n nor vice-versa.

There is some $\ell \in \mathbb{N}$ **with** $f_v(\ell) \geq 2$**:** Let p be the gcd of the set $Z = \{n \in \mathbb{N} \mid f_v(n) \geq 1\}$. If $n \in \mathbb{N}$ is not divisible by p we directly have $f_v(n) = 0$ since this implies $n \notin Z$.

Additionally there is some finite subset $Z' \subseteq Z$ with the same gcd p. By Bézout's lemma there is some $N \in \mathbb{N}$ which is a multiple of p s.t. that every number of the form $N + i \cdot p$ with $i \in \mathbb{N}$ can be written as a non-negative linear combination of elements from Z'. Let $n = N + i \cdot p$ for any $i \in \mathbb{N}$. We can write

$$n = N + \underbrace{\frac{n - N - \lfloor \frac{n-N}{\ell} \rfloor \cdot \ell}{p}}_{\alpha :=} \cdot p + \underbrace{\left\lfloor \frac{n-N}{\ell} \right\rfloor}_{\beta :=} \cdot \ell \,. \tag{4.24}$$

since $\ell \in Z$ implies that p divides ℓ. Furthermore both α and β are non-negative integers and β is maximal with this property. In particular we can write $N + \alpha \cdot p$ as a non-negative linear combination with elements from Z'. This gives $N + \alpha \cdot p = \sum_{p_j \in Z'} \kappa_j \cdot p_j$ for $\kappa_j \in \mathbb{N}$. Putting this together we have

$$\begin{aligned}
f_v(n) &= (A^n)_{vv} \\
&= (A^{N+\alpha\cdot p+\beta\cdot\ell})_{vv} \\
&= (A^{\sum_{p_j\in Z'} \kappa_j\cdot p_j+\beta\cdot\ell})_{vv} \\
&\geq (A^{\beta\cdot\ell})_{vv} \cdot \prod_{p_j \in Z'} (A^{\kappa_j \cdot p_j})_{vv} \\
&\geq (A^{\ell})^{\beta}_{vv} \cdot \prod_{p_j \in Z'} (A^{p_j})^{\kappa_j}_{vv} \\
&= f_v(\ell)^{\beta} \cdot \prod_{p_j \in Z'} f_v(p_j)^{\kappa_j} \\
&\geq f_v(\ell)^{\beta} \\
&\geq 2^{\beta} \\
&\geq 2^{\frac{n-N}{\ell}-1} \in 2^{\Omega(n)}
\end{aligned}$$

Note that no entry in A^n can grow faster than linearly exponential, so this lower bound directly induces a function $g_v \in 2^{\Theta(n)}$ rather than a function $g_v \in 2^{\Omega(n)}$.

In the graph setting this corresponds to repeating different walks from v to v of length ℓ for most of the walk, thus accumulating a linearly exponential amount of different walks and then finishing with some bounded amount of walks from v to v with lengths in Z'.

□

We can then lift this result to all entries of A^n.

4.25 Lemma. *If $A \in \mathbb{N}^{k\times k}$, then each entry of A^n is an ultimately almost PORC function.*

Proof. We use the description of the diagonal entries of the matrices A_S^n given by Lemma 4.23 for all sets $S \subseteq [k]$ where A_S is the matrix A obtained by removing the rows and columns given by S. These correspond to the number of walks of length n from a vertex v to itself in the multigraph associated with the adjacency matrix A, restricted to the induced subgraph spanned by all the vertices not in S. For any $S \subseteq [k]$ we will index A_S the same way as we index A, i.e. $(A_S)_{ij} = A_{ij}$ whenever $i, j \in [k] \setminus S$. Lemma 4.23 in particular already proves this lemma for the diagonal entries.

Let $G = (V, E)$ with $V = [k]$ be the multigraph defined by the adjacency matrix A. Let W be any walk on G. Then we can obtain a unique simple path P_W from W by repeatedly contracting cycles in W. We call two walks W_1 and W_2 equivalent if $P_{W_1} = P_{W_2}$. Note that this forms an equivalence relation on the walks on G and there is a bijection between paths on G and equivalence classes of walks on G and we can construct every walk in the equivalence class $\mathcal{W}_P$ corresponding to a path[9] $P = v_1 \to \ldots \to v_r$ as follows: At every vertex v_i we can add any walk from v_i back to v_i of any length ℓ_i, given that it doesn't pass through any of the vertices $v_1, \ldots, v_{i-1}$. Note that this restriction is not necessary in general, but it ensures that every walk is uniquely constructed, a necessity for our proof. This construction gives a walk of length $r - 1 + \sum_{i=1}^{r} \ell_i$.

[9] A path here is a sequence of edges, so in particular two paths with the same sequence of vertices do not have to be the same since G is a multigraph

Fix some $u \neq v \in V$ and let $P = v_1 \to \ldots \to v_r$ be any path from u to v, i.e. $v_1 = u$ and $v_r = v$. Then the class $\mathcal{W}_P$ contains exactly

$$f_P(n) := \sum_{(\ell_1,\ldots,\ell_r)} \prod_{i=1}^{r} (A_{S_i}^{\ell_i})_{v_i v_i}$$

walks of length $n \in \mathbb{N}$ where the sum is over all $(\ell_1, \ldots, \ell_r) \in \mathbb{N}^r$ with $n = r - 1 + \sum_{i=1}^{r} \ell_i$ and we define $S_i = \{v_1, \ldots, v_{i-1}\}$. By Lemma 4.23 we know the structure of each of the $(A_{S_i}^{\ell_i})_{v_i v_i}$ as a function in ℓ_i. Call the v_i either 0-vertices, 1-vertices or exp-vertices respectively for each of the three cases of Lemma 4.23. Additionally denote by p_i the corresponding period for the 1-vertices and exp-vertices and in case of exp-vertices

denote by N_i the smallest multiple of p_i, s.t. $(A_{S_i}^{N_i+j\cdot p_i})_{v_iv_i} > 0$ for all $j \in \mathbb{N}$. Define $I_0 := \{i \in [r] \mid v_i \text{ is a 0-vertex}\}$ and let I_1 and $I_{\exp}$ be defined analogously.

We first look at the case $I_{\exp} = \emptyset$. Then $\prod_{i=1}^{r}(A_{S_i}^{\ell_i})_{v_iv_i} = 1$ iff $\ell_i = 0$ for all $i \in I_0$ and ℓ_i is a multiple of p_i for all $i \in I_1$, otherwise $\prod_{i=1}^{r}(A_{S_i}^{\ell_i})_{v_iv_i} = 0$. Thus $f_P(n)$ is given as the number of integer points in the polytope defined by $\sum_{i\in I_1} \ell_i \cdot p_i = n - r + 1$ and $\ell_i \geq 0$ for all $i \in I_1$. This is a scaled polytope with rational vertices and thus $f_P(n)$ is the Ehrhart quasi-polynomial (see [BR07, Chapter 3] for an introduction to Ehrhart Theory).

We now look at the case $I_{\exp} \neq \emptyset$. Let v_e be one vertex with $e \in I_{\exp}$ and let p be the gcd of all p_i with $i \in I_1 \cup I_{\exp}$. This implies that $\prod_{i=1}^{r}(A_{S_i}^{\ell_i})_{v_iv_i} = 0$ whenever any of the ℓ_i with $i \in [r]$ are not a multiple of p and thus $f_P(n) = 0$ if $n \not\equiv_p r - 1$. Additionally by Bézout's lemma there is some $N \in \mathbb{N}$ which is a multiple of p s.t. that every number of the form $N + j \cdot p$ with $j \in \mathbb{N}$ can be written as a non-negative linear combination of the p_i with $i \in I_1 \cup I_{\exp}$. Define $N' := N + r - 1 + \sum_{i\in I_{\exp}} N_i$. Let $n = N' + j \cdot p$ for any $j \in \mathbb{N}$. We can write

$$n = N' + \underbrace{\frac{n - N' - \left\lfloor \frac{n-N'}{p_e} \right\rfloor \cdot p_e}{p}}_{\alpha:=} \cdot p + \underbrace{\left\lfloor \frac{n - N'}{p_e} \right\rfloor}_{\beta:=} \cdot p_e \,. \tag{4.26}$$

since $e \in I_{\exp}$ implies that p divides p_e. Furthermore both α and β are non-negative integers and β is maximal with this property. In particular we can write $N + \alpha \cdot p$ as a non-negative linear combination with elements p_i with $i \in I_1 \cup I_{\exp}$. This gives $N' + \alpha \cdot p = r - 1 + \sum_{i\in I_{\exp}} N_i + \sum_{i\in I_1\cup A_{\exp}} \kappa_i \cdot p_i$ for $\kappa_i \in \mathbb{N}$. We choose $\ell_i = 0$ for $i \in I_0$ and $\ell_i = \kappa_i \cdot p_i$ for $i \in I_1$ and $\ell_i = N_i + \kappa_i \cdot p_i$ for $i \in I_{\exp} \setminus \{e\}$ and $\ell_e = N_e + \kappa_e \cdot p_e + \beta \cdot p_e$. This describes a subset of the walks of length

$$r - 1 + \sum_{i\in[r]} \ell_i = r - 1 + \sum_{i\in I_{\exp}} N_i + \sum_{i\in I_1\cup I_{\exp}} \kappa_i \cdot p_i + \beta \cdot p_e = N' + \alpha \cdot p + \beta \cdot p_e = n \,.$$

In particular we have $(A_{S_i}^{\ell_i})_{v_iv_i} \geq 1$ for all $i \in [r]$ and $(A_{S_e}^{\ell_e})_{v_ev_e} \in 2^{\Theta(\ell_e)} = 2^{\Theta(n)}$ and thus $f_P(n) \geq 2^{\Omega(n)}$ for any $n \geq N'$ with $n \equiv_p r - 1$. Note again that $f_P(n)$ can never grow faster than linearly exponential as it is bounded by the total number of walks through G, so $f_P(n)$ is of the form required by the statement of this lemma.

To finish the proof note that the class of ultimately almost PORC functions is closed under addition and multiplication. If we multiply a polynomial constituent with an exponential constituent, then either the polynomial constituent is identical to 0 or there is some $N \in \mathbb{N}$, s.t. the polynomial is strictly positive for every $n \geq N$. We have

$$(A^n)_{uv} = \sum_P f_P(n)$$

where the sum is over all paths P starting at u and ending at v. For any fixed $u, v \in V$ this is a sum of constantly many elements, bounded by $(k \cdot \max(A))^{k-1}$, where $\max(A)$

Figure 4.1: NFAs computing the functions $1^n \mapsto n$ and $1^n \mapsto 2^n$ respectively. Edges with a multiplicity of 2 are denoted by listing the edge label twice.

denotes the maximum entry of A. We conclude that $(A^n)_{uv}$ is an ultimately almost PORC function. □

4.27 Theorem (Classification of univariate functional closure properties of #FA)**.** *A function $\varphi : \mathbb{N} \to \mathbb{N}$ is a functional closure property of #FA iff φ is an ultimately PORC function. This even holds when #FA is restricted to unary languages.*

Proof. Lemma 4.21 already shows that every ultimately PORC function is a closure property of #FA. It remains to show that all functional closure properties of #FA are ultimately PORC functions. For this let $\varphi : \mathbb{N} \to \mathbb{N}$ be a functional closure property of #FA. The function $f : \{1\}^\star \to \mathbb{N}$ defined by $f(1^n) = n$ is computed by the left NFA in Figure 4.1 and thus in #FA. Consequently $\varphi \circ f \in$ #FA. Let $M = (Q, \{1\}, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be an NFA computing $\varphi \circ f$. By Lemma 4.22 this NFA induces a transition matrix $A \in \mathbb{N}^{|Q|\times|Q|}$ and vectors $a, b \in \mathbb{N}^{|Q|}$, s.t. $a^T A^n b = \varphi(f(1^n)) = \varphi(n)$ for all $n \in \mathbb{N}$. Every entry of A^n is an ultimately almost PORC function by Lemma 4.25 and thus $a^T A^n b = \varphi(n)$ is one as well. Let p be the quasiperiod of φ and let φ_i be one of the constituents of φ corresponding to some residue class $i \in \{0, \ldots, p-1\}$. Assume for the sake of contradiction that φ_i grows in $2^{\Theta(n)}$, i.e. there is a constant $\gamma \in \mathbb{R}^+$ and $N \in \mathbb{N}$, s.t. for every $n \geq N$ we have $\varphi_i(n) \geq 2^{\gamma n}$. Consider the function $f_{p,i} : \{1\}^\star \to \mathbb{N}$ defined by $f_{p,i}(1^n) = p \cdot (2^n + N) + i$. We claim $f_{p,i}$ is in #FA. The function $1^n \mapsto 2^n$ is computed by the right NFA in Figure 4.1 and thus in #FA. The remainder of the claim follows by Lemma 4.4 and Lemma 4.5. Note that $f_{p,i}(1^n) \equiv_p i$, so $\varphi \circ f_{p,i} = \varphi_i \circ f_{p,i}$ has to be in #FA as well. However $\varphi(f_{p,i}(1^n)) = \varphi_i(f_{p,i}(1^n)) = \varphi_i(p \cdot (2^n + N) + i) \geq 2^{\gamma p \cdot (2^n+N)+\gamma i}$ for all $n \in \mathbb{N}$ which is larger than any NFA can compute, since NFAs can only compute functions that are at most linearly exponential in the length of the input. We conclude that none of the constituents φ_i of φ can be exponential, so they are instead all polynomials, making φ an ultimately PORC function. □

4.3.2 Multivariate Functional Closure Properties

4.28 Theorem (Classification of multivariate functional closure properties of #FA)**.** *A function $\varphi : \mathbb{N}^m \to \mathbb{N}$ is a functional closure property of #FA iff φ can be written as a finite sum of finite products of univariate ultimately PORC functions.*

Proof. By Theorem 4.27 any univariate ultimately PORC function is a closure property of #FA. As such any finite sum or finite product of them is also a closure property of #FA by Lemmas 4.4 and 4.5.

It remains to show that all functional closure properties of #FA are of this form. For this let $\varphi : \mathbb{N}^m \to \mathbb{N}$ be a functional closure property of #FA. Define the alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ and the functions $f_i : \Sigma^\star \to \mathbb{N}$ where $f_i(w) := \#_i(w)$ is defined as the number of occurences $\#_i(w)$ of the symbol σ_i in w. Applying the closure property to $f_1, \ldots, f_m$ gives that $\varphi \circ (f_1, \ldots, f_m) \in \#\mathsf{FA}$ and thus is computed by an NFA $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$. This induces transition matrices $A_\sigma \in \mathbb{N}^{|Q|\times|Q|}$ for each symbol $\sigma \in \Sigma$ and vectors $a, b \in \mathbb{N}^{|Q|}$, s.t. $a^T \prod_{j=1}^{|w|} A_{w_j} b = \varphi(f_1(w), \ldots, f_m(w))$ for all $w \in \Sigma^\star$. Restricting to words of the form $w = \sigma_1^{n_1}\sigma_2^{n_2}\cdots\sigma_m^{n_m}$ for $n_1, \ldots, n_m \in \mathbb{N}$ gives $a^T \left(\prod_{i=1}^m A_{\sigma_i}^{n_i}\right) b = \varphi(n_1, \ldots, n_m)$. Using Lemma 4.25 on each of the $A_{\sigma_i}^{n_i}$ we see that every entry of $A_{\sigma_i}^{n_i}$ is an ultimately almost PORC function in n_i. Consequently, every entry of $\prod_{i=1}^m A_{\sigma_i}^{n_i}$ is a finite sum of products of different ultimately almost PORC functions and the same holds for $a^T \left(\prod_{i=1}^m A_{\sigma_i}^{n_i}\right) b = \varphi(n_1, \ldots, n_m)$.

We now look at the individual summands of φ and prove that we can rewrite each one as a product of ultimately PORC functions by one-by-one rewriting the exponential constituents. For this let $\varphi^{(1)}(n_1)\cdots\varphi^{(m)}(n_m)$ be one of the summands of φ where $\varphi^{(1)}, \ldots, \varphi^{(m)}$ are all ultimately almost PORC functions, with periods $p_1, \ldots, p_m$, offsets $N_1, \ldots, N_m$ and constituents $\varphi_0^{(i)}, \ldots, \varphi_{p_i-1}^{(i)}$ for each $i \in [m]$. If none of the constituents are exponential we are done. Otherwise let $\varphi_j^{(i)}$ be one of the exponential constituents, let $\gamma \in \mathbb{R}^+$ and let $N \in \mathbb{N}$, s.t. $\varphi_j^{(i)}(n_i) \geq 2^{\gamma n_i}$ for $n_i \geq N$. We claim we can set $\varphi_j^{(i)}(n_i) = 0$ without changing the product $\varphi^{(1)}(n_1)\cdots\varphi^{(m)}(n_m)$ for any $n_1, \ldots, n_m \in \mathbb{N}$. Call the resulting functions $\psi^{(i)}$ and $\psi_j^{(i)}$. Assume for the sake of contradiction, that there are some $c_1, \ldots, c_m \in \mathbb{N}$ where

$$\varphi^{(1)}(c_1)\cdots\varphi^{(m)}(c_m) \neq \varphi^{(1)}(c_1)\cdots\varphi^{(i-1)}(c_{i-1})\cdot\psi^{(i)}(c_i)\cdot\varphi^{(i+1)}(c_{i+1})\cdots\varphi^{(m)}(c_m)\,.$$

This implies that $\varphi^{(1)}(c_1)\cdots\varphi^{(i-1)}(c_{i-1})\cdot\varphi^{(i+1)}(c_{i+1})\cdots\varphi^{(m)}(c_m) \neq 0$ and $c_i \geq N_i$ as we didn't change any other functions except $\varphi^{(i)}$ for $n_i \geq N_i$.

Construct the constant functions $f_k'(w) = c_k$ for $k \neq i$ and the function $f_i'(w) = p_i \cdot (2^{|w|} + \max(N_i, N)) + j$ which are all in #FA. We see that $\varphi \circ (f_1', \ldots, f_m') \in \#\mathsf{FA}$. Note that for any $w \in \Sigma^\star$ we have $f_i'(w) \geq \max(N_i, N)$ and $f_i'(w) \equiv_{p_i} j$ and thus $\varphi^{(i)} \circ f_i' = \psi_j^{(i)} \circ f_i'$. Combining all of this we again reach a contradiction to the fact that NFAs can only compute at most linearly exponential functions via

$$\begin{aligned}\varphi(f'_1(w),\dots,f'_m(w)) &\geq \varphi^{(1)}(c_1)\cdots\varphi^{(i-1)}(c_{i-1})\cdot\varphi^{(i)}(f'_i(w))\cdot\varphi^{(i+1)}(c_{i+1})\cdots\varphi^{(m)}(c_m)\\ &\geq \varphi^{(i)}(f'_i(w)) \;=\; \varphi^{(i)}_j(f'_i(w)) \;=\; \varphi^{(i)}_j(p_i\cdot(2^{|w|}+\max(N_i,N))+j)\\ &\geq 2^{\gamma p_i\cdot(2^{|w|}+\max(N_i,N))+\gamma j}\,.\end{aligned}$$

Note that the first inequality holds due to all summands of φ being non-negative. □

Deciding whether a function φ has such a representation may not always be directly visible, however if φ is a multivariate polynomial we can be more explicit. Every integer-valued multivariate polynomial has integer coefficients when represented in the binomial basis (see [IP22, Prop. 4.2.1] for a proof of this fact). We say a term $a\cdot\binom{x_1}{d_1}\cdots\binom{x_m}{d_m}$ dominates another term $a'\cdot\binom{x_1}{d'_1}\cdots\binom{x_m}{d'_m}$ if $d_i\geq d'_i$ for all $i\in[m]$. A term is a dominating term of φ if it has a non-zero coefficient and it is not dominated by any other term with non-zero coefficient. We can use a similar approach to Lemma 4.19 to rewrite φ as a positive integer linear combination of products of shifted binomials.

4.29 Lemma. *Let $\varphi(x_1,\dots,x_m)=\sum_{i=1}^{r}a_i\cdot\prod_{j=1}^{m}\binom{x_j}{d_{i,j}}$ with $a_i\in\mathbb{Z}$ and the coefficients of the dominating terms being positive. Then there are $a'_1,\dots,a'_{r'}\in\mathbb{N}$ and $c_1,\dots,c_{r'}\in\mathbb{N}$ with $\varphi(x_1,\dots,x_m)=\sum_{i=1}^{r'}a'_i\cdot\prod_{j=1}^{m}\binom{x_j-c_i}{d_{i,j}}$.*

Proof. Let D_φ be the set of dominating terms of φ. We consider a set of dominating terms D to be smaller than another set of dominating terms D', whenever $D\neq D'$ and all terms in D are dominated by terms in D'. We prove the statement via induction over D. In case $D_\varphi=\emptyset$ we have $\varphi=0$ and are done. So let $D_\varphi\neq\emptyset$ and let $a\cdot\binom{x_1}{d_1}\cdots\binom{x_m}{d_m}$ be any term from D_φ. We use a similar idea to Lemma 4.19 and replace the binomial basis by a shifted binomial basis. Consider the terms $a_i\cdot\binom{x_1}{d_1}\cdots\binom{x_{i-1}}{d_{i-1}}\binom{x_i}{d_i-1}\binom{x_{i+1}}{d_{i+1}}\cdots\binom{x_m}{d_m}$ in φ for all $i\in[m]$ with $d_i\geq 1$. These are precisely the potential new dominating terms of $\varphi-a\cdot\binom{x_1}{d_1}\cdots\binom{x_m}{d_m}$. In order to ensure the positivity of the a_i we instead consider $\psi:=\varphi-a\cdot\binom{x_1-c}{d_1}\cdots\binom{x_m-c}{d_m}$ for some $c\in\mathbb{N}$ with $c>-\frac{a_i}{a}$ for all $i\in[m]$ with $d_i\geq 1$. We recall that the Chu-Vandermonde identity gives us $\binom{x-c}{r}=\sum_{i=0}^{r}(-1)^{r-i}\binom{r-i+c-1}{r-i}\binom{x}{i}$. Thus the new coefficient a'_i of $\binom{x_1}{d_1}\cdots\binom{x_{i-1}}{d_{i-1}}\binom{x_i}{d_i-1}\binom{x_{i+1}}{d_{i+1}}\cdots\binom{x_m}{d_m}$ in ψ is $a_i+a\cdot(-c)>0$. The new coefficient of $\binom{x_1}{d_1}\cdots\binom{x_m}{d_m}$ in ψ is zero and thus D_ψ is smaller than D_φ. By induction we know that ψ can be written as a positive linear combination of products of shifted binomials and thus $\varphi=\psi+a\cdot\binom{x_1-c}{d_1}\cdots\binom{x_m-c}{d_m}$ has the same property. □

We can then use a generalization of Lemma 4.20 and replace subtractions x_i-c' by $\max(x_i-c',0)$. However special care has to be taken for $x_i<c'$, in which case x_i has to be replaced by the corresponding constants first. This adds the additional condition on φ.

4.30 Lemma. *Let $\varphi : \mathbb{N}^m \to \mathbb{N}$ be a multivariate polynomial with rational coefficients, such that whenever ψ is formed from φ by replacing any set of variables – including the empty subset – by constants from $\mathbb{N}$, then all dominating terms of ψ have positive coefficients. Then φ is a functional closure property of $\#\mathsf{FA}$.*

Proof. We prove the statement via induction over the number of variables m. If $m \leq 1$, then φ is a constant polynomial and thus a functional closure property of $\#\mathsf{FA}$ by Lemma 4.20. For $m \geq 2$, let $f_1, \ldots, f_m \in \#\mathsf{FA}$. Write φ as a positive integer linear combination of products of shifted binomials using Lemma 4.29 and let $c \in \mathbb{N}$ be the maximum occurring shift. We now distinguish between the f_i taking on values from $\{0, \ldots, c-1\}$ and them taking on values of at least c and all combinations thereof for the different f_i. Using the fact that all subtractions subtract at most a value of c we can replace the subtractions $x_i - c'$ by $\max(x_i - c', 0)$ without changing the value for $x_i \geq c$ and get the following:

$$\varphi(f_1(w), \ldots, f_m(w)) = \Big(\prod_{i\in[m]} \mathbb{1}_{f_i(w)\geq c}\Big) \cdot \varphi_{\text{clamped}}(f_1(w), \ldots, f_m(w)) + \sum_{\emptyset\subsetneq I\subseteq[m]} \sum_{\rho} \Big(\prod_{i\in I} \mathbb{1}_{f_i(w)=\rho(i)}\Big) \cdot \Big(\prod_{i\in[m]\setminus I} \mathbb{1}_{f_i(w)\geq c}\Big) \cdot \varphi_{I,\rho}(f_1(w), \ldots, f_m(w))$$

where ρ sums over all functions $I \to \{0, \ldots, c-1\}$, φ_{clamped} is φ, but with all subtractions $x_i - c'$ replaced by $\max(x_i - c', 0)$ and $\varphi_{I,\rho}$ is φ, but all variables x_i with $i \in I$ are replaced by the constants $\rho(i)$. The $\varphi_{I,\rho}$ are non-negative integer-valued multivariate polynomials in fewer variables and when replacing any variables by further constants we only obtain polynomials that can also be obtained by replacing variables directly in φ. Thus by induction the $\varphi_{I,\rho}$ are a functional closure property of $\#\mathsf{FA}$ and $\varphi(f_1, \ldots, f_m) \in \#\mathsf{FA}$ by Lemmas 4.4, 4.5, 4.12, 4.14, and 4.18. □

Indeed, this property precisely characterizes the multivariate polynomials that are functional closure properties of $\#\mathsf{FA}$.

4.31 Lemma. *A multivariate polynomial $\varphi : \mathbb{N}^m \to \mathbb{N}$ with rational coefficients is a functional closure property of $\#\mathsf{FA}$ iff for every ψ that can be formed from φ by replacing any subset of variables – including the empty set – by constants from $\mathbb{N}$, then all dominating terms of ψ have positive coefficients.*

Proof. Lemma 4.30 already proves that all multivariate polynomials of this form are functional closure properties of $\#\mathsf{FA}$.

Now let $\varphi : \mathbb{N}^m \to \mathbb{N}$ be a multivariate polynomial with rational coefficient and a functional closure property of $\#\mathsf{FA}$. By Theorem 4.28 φ can be written as a finite sum of finite products of ultimately PORC functions. W.l.o.g. let all quasiperiods be the same $p \in \mathbb{N}$ and all offsets be the same $N \in \mathbb{N}$ and a multiple of p, for example by choosing

the lowest common multiple of the quasiperiods and choosing the lowest multiple of p that is bigger than all offsets. Then

$$\varphi(x_1, \ldots, x_m) = \sum_{i=1}^{r} \prod_{j=1}^{m} \varphi^{(i,j)}(x_j) \tag{4.32}$$

where $\varphi^{(i,j)}$ is an ultimately PORC function with constituents $\varphi_0^{(i,j)}, \ldots, \varphi_{p-1}^{(i,j)}$. Consider inputs on a grid of the form $(N + i_1 \cdot p, N + i_2 \cdot p, \ldots, N + i_m \cdot p)$ for $i_1, \ldots, i_m \in \mathbb{N}$. For all these inputs each of the ultimately PORC functions always uses the same constituents, namely the $\varphi_0^{(i,j)}$. Thus φ agrees with $\varphi'(x_1, \ldots, x_m) := \sum_{i=1}^{r} \prod_{j=1}^{m} \varphi_0^{(i,j)}(x_j)$ on all inputs from this grid. Furthermore the grid is infinite in all dimensions and thus $\varphi = \varphi'$ by multivariate polynomial interpolation (see [GS00] for an exposition of multivariate polynomial interpolation).

Remains to look at the dominating terms: W.l.o.g. assume that each $\varphi_0^{(i,j)}$ is not the constant zero function. Then it has a positive leading coefficient, otherwise $\varphi^{(i,j)}$ would not be a PORC function. Combined with the fact that each of the $\varphi_0^{(i,j)}$ is univariate, the product $\prod_{j=1}^{m} \varphi_0^{(i,j)}(x_j)$ for a fixed $i \in [r]$ has exactly one dominating term when written in the binomial basis. Its coefficient is precisely the product of the leading coefficients of the $\varphi_0^{(i,j)}$ and thus positive. As a result all dominating terms in the binomial basis of φ' (and thus φ) have positive coefficients, since no cancellations can occur unless the dominating term of a summand dominates the dominating term of another summand, in which case the smaller term would no longer be a dominating term.

Note that whenever any variable, for example x_j, in φ is replaced by a constant c, then this only changes each function $\varphi^{(i,j)}(x_j)$ and thus $\varphi_0^{(i,j)}(x_j)$ to be the constant $\varphi^{(i,j)}(c)$ which is non-negative. □

4.4 Promise Closure Properties

As already mentioned in the introduction, $(f_1 - f_2)^2$ can be proven to not be a functional closure property of #FA using the Pumping Lemma. However, what happens if f_1 and f_2 are not entirely arbitrary, but are potentially related, for example if we know $f_1 = 3 \cdot f_2$? Under this assumption, we can rewrite $(f_1 - f_2)^2$ as $4f_2^2$, clearly a functional closure property of #FA. To answer this type of questions we introduce the concept of functional promise closure properties:

4.33 Definition. Let $S \subseteq \mathbb{N}^m$ and let $\varphi : \mathbb{N}^m \to \mathbb{N}$ be a function. We call φ a *functional promise closure property* of #FA with regard to S if for every $f_1, \ldots, f_m \in \#\mathsf{FA}$ defined on some shared alphabet Σ there is a function $g \in \#\mathsf{FA}$ with $g(w) = \varphi(f_1(w), \ldots, f_m(w))$ for every $w \in \Sigma^\star$ with $(f_1(w), \ldots, f_m(w)) \in S$.

While one could also look at a variation where we are guaranteed $(f_1(w), \dots, f_m(w)) \in S$ for every $w \in \Sigma^\star$, it is natural to phrase these requirements as a promise problem: NFAs are unable to even verify many easy constraints, like $f_1 = 3 \cdot f_2$ from our example.

We now want to show that, if S fulfills some property, namely admitting polynomial cluster sequences (we postpone the definition to Definition 4.37), then for every functional promise closure property with regard to S there is a functional closure property of #FA that agrees with it on all tuples in S. In other words we can interpolate the functional promise closure property φ on all values of S to obtain a functional closure property φ' for all of #FA. The proof for this follows along the following ideas: First, similar to Theorem 4.28, use the functional closure property φ on the unary counting functions to find an equivalent function φ' that is almost a functional closure property. However after this step some of the constituents of the ultimately almost PORC functions might still be exponential. Theorem 4.28 then proceeded by showing that we can replace all these exponential constituents by the constant zero function, as otherwise we were able to reach a contradiction by constructing functions $f'_1, \dots, f'_m$ and an infinite sequence of inputs $w^{(i)}$, such that $\varphi(f'_1(w^{(i)}), \dots, f'_m(w^{(i)}))$ grows doubly exponential in the length of the inputs. However when dealing with functional promise closure properties we have to be more careful when choosing $f'_1, \dots, f'_m$ and the $w^{(i)}$, because we need $(f'_1(w^{(i)}), \dots, f'_m(w^{(i)})) \in S$ to reach a contradiction. Additionally we don't replace the exponential constituents by the constant zero-function but rather a polynomial that behaves the same for small inputs. For this we need a special variant of univariate polynomial interpolation that yields integer-valued polynomials that are non-negative for all inputs from $\mathbb{N}$:

4.34 Lemma. *Let $c_0, \dots, c_N \in \mathbb{N}$. Then there is an integer-valued polynomial $q : \mathbb{Q} \to \mathbb{Q}$ with $q(n) = c_n$ for all $n \in \{0, \dots, N\}$ and $q(n') \geq 0$ for all $n' \in \mathbb{N}$.*

Proof. We inductively construct integer-valued polynomials $q_0, \dots, q_N$, that each interpolate one value more. We start by setting $q_0(x) := c_0 \cdot \binom{x}{0}$. Clearly $q_0(0) = c_0$. We proceed inductively by setting $q_{i+1}(x) := q_i(x) + (c_{i+1} - q_i(i+1)) \cdot \binom{x}{i+1}$. Note that $\binom{n}{i+1} = 0$ for $n \in \{0, \dots, i\}$ and thus q_{i+1} and q_i agree on all such n. The final q is then constructed as $q(x) := q_N(x) + \alpha \cdot \binom{x}{N+1}$, for some big enough $\alpha \in \mathbb{N}$, s.t. $q(n') \geq 0$ for all $n' \in \mathbb{N}$. Such an α always exists since $\binom{x}{N+1}$ has higher degree than q_N. □

To be able to hit all of S consistently we use independent binary encodings, that allow us to hit all of $\mathbb{N}^m$.

4.35 Lemma (Folklore)**.** *The function $f : \{0,1\}^\star \to \mathbb{N}$ defined by being the value of $w \in \{0,1\}^\star$ interpreted as a binary number is in #FA. Additionally it is possible to extend the domain of f to any alphabet $\Sigma \supseteq \{0,1\}$ where the value of f is determined while ignoring any symbols not in $\{0,1\}$.*

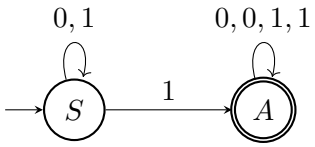


Figure 4.2: NFA computing the binary the value of its input interpreted as a binary number. Edges with a multiplicity of 2 are denoted by listing the edge label twice.

Proof. We define

$$a = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad A_\sigma = \begin{pmatrix} 1 & \sigma \\ 0 & 2 \end{pmatrix}, \qquad \text{and} \qquad b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

A visualization of the corresponding NFA M via Lemma 4.22 can be found in Figure 4.2. We prove that M computes f by showing via induction that for every $w \in \{0,1\}^\star$ with $|w| = n$ we have

$$\prod_{i=1}^{n} A_{w_i} = \begin{pmatrix} 1 & f(w) \\ 0 & 2^n \end{pmatrix}.$$

For $n = 0$ this is obviously correct as $f(\varepsilon) = 0$. For the induction step consider the word $w\sigma$ of length $n+1$ for $\sigma \in \{0,1\}$. Then

$$\left(\prod_{i=1}^{n} A_{w_i}\right) \cdot A_\sigma = \begin{pmatrix} 1 & f(w) \\ 0 & 2^n \end{pmatrix} = \begin{pmatrix} 1 & \sigma + 2f(w) \\ 0 & 2^{n+1} \end{pmatrix} = \begin{pmatrix} 1 & f(w\sigma) \\ 0 & 2^{n+1} \end{pmatrix}$$

and we are done. Extending the alphabet and ignoring those new symbols can be done by adding a self-loop to each state of M. This corresponds to an identity matrix and thus doesn't influence the result. □

For any $n \in \mathbb{N}$ we denote by $\mathrm{bin}(n)$ the unique binary representation of n without leading zeros. We first show the methodology in detail by proving the univariate case.

4.36 Theorem. *Let $S \subseteq \mathbb{N}$. Then any function $\varphi : \mathbb{N} \to \mathbb{N}$ is a functional promise closure property of #FA with regard to S iff there is a functional closure property $\psi : \mathbb{N} \to \mathbb{N}$ of #FA with $\varphi_{|S} = \psi_{|S}$.*

Proof. If such a ψ exists, we directly see that φ is a functional promise closure property of #FA with respect to S. Indeed for any $f \in \#\mathsf{FA}$ we construct $g = \psi \circ f \in \#\mathsf{FA}$ and see $g(w) = \varphi(f(w))$ for every $w \in \Sigma^\star$ with $f(w) \in S$.

Now on the other hand let φ be any functional promise closure property of #FA with regard to S. Again define the alphabet $\Sigma = \{1\}$ and the function $f : \Sigma^\star \to \mathbb{N}$ with

$f(1^n) := n$. Applying the closure property to f gives that there is some $g \in \#\mathsf{FA}$ with $g(1^n) = \varphi(f(1^n)) = \varphi(n)$ for all $n \in S$. Let $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be an NFA computing g. This induces a transition matrix $A \in \mathbb{N}^{|Q|\times|Q|}$ and vectors $a, b \in \mathbb{N}^{|Q|}$, s.t. $a^T A^n b = g(1^n)$ for all $n \in \mathbb{N}$ by Lemma 4.22. We now define $\chi(n) := a^T A^n b$ and see $\chi_{|S} = \varphi_{|S}$. Using Lemma 4.25 on A^n we see that every entry of A^n is an ultimately almost PORC function in n and as such ψ is also an ultimately almost PORC function. Let p be the quasiperiod of χ and let $\chi_0, \dots, \chi_{p-1}$ be the constituents of χ and let $N \in \mathbb{N}$ be the offset after which χ is defined by the constituents.

We claim that we can now replace every exponential constituent by a polynomial without changing the value of χ for any $n \in S$. For each $i \in \{0, \dots, p-1\}$ we distinguish two cases, depending on whether $S_i := S \cap (p\mathbb{Z} + i)$ is finite or it is infinite. If S_i is a finite set, we replace χ_i' with a polynomial that interpolates the same values as χ_i on S_i. Lemma 4.34 ensures that this polynomial is integer-valued and non-negative for all of $\mathbb{N}$. If S_i is an infinite set, we replace χ_i by the constant zero function. Call the resulting ultimately PORC function ψ with constituents ψ_i. Assume for the sake of contradiction there is a $c \in S$, s.t. $\chi(c) \neq \psi(c)$. Clearly such a c would have to be at least N. Now let i be, s.t. $c \in S_i$. It must hold that $\chi_i(c) \neq \psi_i(c)$. Hence S_i cannot be a finite set, since χ_i and ψ_i agree on S_i. Therefore S_i must be infinite. Since χ_i is exponential there is a $\gamma \in \mathbb{R}^+$ and $N' \in \mathbb{N}$, s.t. $\chi_i(n) \geq 2^{\gamma n}$ for all $n \geq N'$. Let $f' \in \#\mathsf{FA}$ be the function of binary evaluation from Lemma 4.35 over the alphabet $\Sigma' = \{0, 1\}$. Then there is a $g' \in \#\mathsf{FA}$ with $g'(\mathrm{bin}(n)) = \varphi(f'(\mathrm{bin}(n))) = \chi(f'(\mathrm{bin}(n)))$ for all $n \in S$. Since S_i is infinite, in particular S_i must contain infinitely many values bigger than $\max(N, N')$. For $n \in S_i$ with $n \geq \max(N, N')$ we now have

$$g'(\mathrm{bin}(n)) = \chi(f'(\mathrm{bin}(n))) = \chi(n) = \chi_i(n) \geq 2^{\gamma n} \geq 2^{\gamma 2^{|\,\mathrm{bin}(n)|-1}},$$

which is a contradiction to NFAs only being able to only compute functions that are at most linearly exponential in the input length. In conclusion S_i cannot be infinite either and thus c itself cannot exist. □

4.37 Definition. A set $S \subseteq \mathbb{N}^m$ admits *polynomial cluster sequences* if for every $N_1, \dots, N_m \in \mathbb{N}$, and $p_1, \dots, p_m \in \mathbb{N}$ the projection $\tau : S \to \{0, \dots, N_1 + p_1 - 1\} \times \dots \times \{0, \dots, N_m + p_m - 1\}$ defined by $\tau(n_1, \dots n_m) = (n_1 \operatorname{srem}_{N_1} p_1, \dots, n_m \operatorname{srem}_{N_m} p_m)$ has the following property: Any preimage T of a singleton set under τ for every $i \in [m]$ has either bounded i-th coordinate or there is a polynomial $q : \mathbb{N} \to \mathbb{N}$ and an infinite subset $T' \subseteq T$ with unbounded i-th coordinate[10] and with $\sum_{j=1}^{m} n_j \leq q(n_i)$ for all $(n_1, \dots, n_m) \in T'$. We call such an infinite subset a *polynomial cluster sequence* with regards to dimension i.

We call τ the *shifted grid projection* of S with respect to offsets $N_1, \dots, N_m$ and quasiperiods $p_1, \dots, p_m$.

[10] note that this property also follows directly from the polynomial bound on the other coordinates in the i-th coordinate, but we have it as part of the definition for clarity.

Intuitively this definition requires that each dimension is either bounded or can grow reasonably quickly together with the other dimensions, even when restricted to inputs from specific shifted residue classes. For example $\{(n^2, n^3) \mid n \in \mathbb{N}\}$ admits polynomial cluster sequences, while $\{(n, 2^n) \mid n \in \mathbb{N}\}$ does not.

4.38 Theorem. *Let $S \subseteq \mathbb{N}^m$ admit polynomial cluster sequences. Then any function $\varphi : \mathbb{N}^m \to \mathbb{N}$ is a functional promise closure property of #FA with regard to S iff there is a functional closure property $\psi : \mathbb{N}^m \to \mathbb{N}$ of #FA with $\varphi_{|S} = \psi_{|S}$.*

Proof. If such a ψ exists, we directly see that φ is a functional promise closure property of #FA with respect to S. Indeed for any $f_1, \ldots, f_m \in \#\mathsf{FA}$ we construct $g = \psi \circ (f_1, \ldots, f_m) \in \#\mathsf{FA}$ and see $g(w) = \varphi(f_1(w), \ldots, f_m(w))$ for every $w \in \Sigma^\star$ with $(f_1(w), \ldots, f_m(w)) \in S$.

Now on the other hand let φ be any functional promise closure property of #FA with regard to S. Again define the alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ and the functions $f_i : \Sigma^\star \to \mathbb{N}$ where $f_i(w) := \#_i(w)$ is defined as the number of occurences $\#_i(w)$ of the symbol σ_i in w. Applying the closure property to $f_1, \ldots, f_m$ gives that there is some $g \in \#\mathsf{FA}$ with $g(w) = \varphi(f_1(w), \ldots, f_m(w))$ for all $w \in \Sigma^\star$ with $(f_1(w), \ldots, f_m(w)) \in S$. Let $M = (Q, \Sigma, \mathrm{wt}, \mathrm{in}, \mathrm{out})$ be an NFA computing g. This induces transition matrices $A_\sigma \in \mathbb{N}^{|Q| \times |Q|}$ for each symbol $\sigma \in \Sigma$ and vectors $a, b \in \mathbb{N}^{|Q|}$, s.t. $a^T \prod_{j=1}^{|w|} A_{w_j} b = g(x)$ for all $w \in \Sigma^\star$. Restricting to words of the form $w = \sigma_1^{n_1} \sigma_2^{n_2} \cdots \sigma_m^{n_m}$ for $n_1, \ldots, n_m \in S$ gives $a^T \left(\prod_{i=1}^m A_{\sigma_i}^{n_i}\right) b = \varphi(n_1, \ldots, n_m)$. Using Lemma 4.25 on each of the $A_i^{n_i}$ we see that every entry of $A_i^{n_i}$ is an ultimately almost PORC function in n_i. Consequently every entry of $\prod_{i=1}^m A_{\sigma_i}^{n_i}$ is a finite sum of products of different ultimately almost PORC functions and the same holds for $\chi(n_1, \ldots, n_m) := a^T \left(\prod_{i=1}^m A_{\sigma_i}^{n_i}\right) b$. Note that $\chi_{|S} = \varphi_{|S}$.

Remains to show that we can rewrite χ as some ψ with $\psi_{|S} = \chi_{|S}$, s.t. none of the constituents are exponential. Again we do this by looking at each summand individually and replacing one exponential constituent at a time. For this let $\chi^{(1)}(n_1) \cdots \chi^{(m)}(n_m)$ be one of the summands of χ where $\chi^{(1)}, \ldots, \chi^{(m)}$ are all ultimately almost PORC functions, with periods $p_1, \ldots, p_m$, offsets $N_1, \ldots, N_m$ and constituents $\chi_0^{(i)}, \ldots, \chi_{p_i-1}^{(i)}$ for each $i \in [m]$. W.l.o.g. choose the offsets big enough, such that every constituent $\chi_j^{(i)}$ is either constant zero for $n_i \geq N_i$ or never zero for any $n_i \geq N_i$. Let $\chi_j^{(i)}$ be one of the exponential constituents and let $\gamma \in \mathbb{R}^+$ and $N \in \mathbb{N}$ be, s.t. $\chi_j^{(i)}(n_i) \geq 2^{\gamma n_i}$ for $n_i \geq N$. Let $B_{\max,i}$ be the maximum the i-th coordinate assumes in any preimage of a singleton set under τ where the i-th coordinate is bounded. We replace $\chi_j^{(i)}$ with a polynomial that interpolates $\chi^{(i)}$ for all values of at most $B_{\max,i}$. Lemma 4.34 ensures this is possible with an integer-valued polynomial that is non-negative for all of $\mathbb{N}$. Call the resulting functions after replacement $\psi^{(i)}$ and $\psi_j^{(i)}$. We claim that this does not affect the value of the product for any $(n_1, \ldots, n_m) \in S$. For the sake of contradiction assume the value of the product did change for some $(c_1, \ldots, c_m) \in S$. We thus have

$$\chi^{(1)}(c_1) \cdots \chi^{(m)}(c_m) \neq \chi^{(1)}(c_1) \cdots \chi^{(i-1)}(c_{i-1}) \cdot \psi^{(i)}(c_i) \cdot \chi^{(i+1)}(c_{i+1}) \cdots \chi^{(m)}(c_m)$$

which implies $\chi^{(1)}(c_1)\cdots\chi^{(i-1)}(c_{i-1})\cdot\chi^{(i+1)}(c_{i+1})\cdots\chi^{(m)}(c_m) \neq 0$ and $c_i \geq N_i$ as no function except $\chi^{(i)}$ for inputs at least as big as N_i has been changed. By the same reason we know $\chi_j^{(i)}(c_i) \neq \psi_j^{(i)}(c_i)$ and $c_i \equiv_{p_i} j$.

Consider the preimage $T = \tau^{-1}(c_1 \operatorname{srem}_{N_1} p_1, \ldots, c_m \operatorname{srem}_{N_m} p_m)$. Clearly $(c_1, \ldots, c_m) \in T$. The i-th coordinate of T has to be unbounded, because $c_i \leq B_{\max,i}$ would imply $\psi_j^{(i)}(c_i) = \chi^{(i)}(c_i) = \chi_j^{(i)}(c_i)$ by the construction of $\psi_j^{(i)}$. Furthermore the behavior of each of the $\chi^{(k)}$ is only dictated by at most one of the constituents each. By our choice of the offsets we have $\chi^{(k)}(n_k) \neq 0$ for all $k \neq i$ and all $(n_1, \ldots, n_m) \in T$. Since S admits polynomial cluster sequences there is a polynomial cluster sequence $T' \subseteq T$ with regards to dimension i with polynomial $q : \mathbb{N} \to \mathbb{N}$.

We construct m functions $f'_k \in \#\mathsf{FA}$ over the alphabet $\Sigma = \{\sigma_{1,0}, \sigma_{1,1}, \ldots, \sigma_{m,0}, \sigma_{m,1}\}$ with $2m$ symbols, by using Lemma 4.35 on the symbols $\sigma_{k,0}$ and $\sigma_{k,1}$ each while ignoring any of the other symbols. This allows us to encode any $(n_1, \ldots, n_m) \in T'$ as independent binary representations without leading zeros for each n_k into a string of total length $\sum_{k=1}^{m} \lfloor \log_2(n_k) + 1 \rfloor \leq m \cdot \lfloor \log_2(q(n_i)) + 1 \rfloor$ which is in $O(\log(n_i))$ for fixed m and q.

Now let $w \in \Sigma^\star$ be any such string corresponding to a value $(n_1, \ldots, n_m) \in T'$. Combining all of this we again reach a contradiction to the fact that NFAs can only compute at most linearly exponential functions via

$$\begin{aligned}
\varphi(f'_1(w), \ldots, f'_m(w)) &\geq \chi^{(1)}(c_1)\cdots\chi^{(i-1)}(c_{i-1})\cdot\chi^{(i)}(f'_i(w))\cdot\chi^{(i+1)}(c_{i+1})\cdots\chi^{(m)}(c_m) \\
&\geq \chi^{(i)}(f'_i(w)) \\
&= \chi_j^{(i)}(f'_i(w)) \\
&= \chi_j^{(i)}(n_i) \\
&\geq 2^{\gamma n_i}\,.
\end{aligned}$$

Note that $|w| \in O(\log(n_i))$ implies $2^{\gamma n_i}$ is doubly exponential in the input length. □

There are multiple natural families for the set S such that S admits polynomial cluster sequences which we describe in the following.

4.39 Lemma. *Any finite set $S \subseteq \mathbb{N}^m$ admits polynomial cluster sequences.*

Proof. Independent of the offsets, quasiperiods, and $i \in [m]$, all subsets of S are finite and thus bounded in every dimension. □

An affine variety is defined as the zero set of a finite number of multivariate polynomials. A special case of affine varieties are *graph varieties* (also just called *graphs*, see [Sha13,

§2.4, Exe. 12]). An affine variety S is a graph variety if there exist a finite number of j-variate polynomials $\mu_1, \ldots, \mu_k$ such that

$$S = \{(s_1, \ldots, s_j, \mu_1(s_1, \ldots, s_j), \ldots, \mu_k(s_1, \ldots, s_j)) \mid (s_1, \ldots, s_j) \in \mathbb{Q}^j\} \subseteq \mathbb{Q}^{j+k}\,.$$

We call $s_1, \ldots, s_j$ the *free* variables, and the remaining variables the *dependent* variables. We call S a monotone graph variety if $\mu_1, \ldots, \mu_k$ are all monotone.

4.40 Lemma. *Let $S \subseteq \mathbb{Q}^m = \mathbb{Q}^{j+k}$ be a monotone graph variety. Then the set $S \cap \mathbb{N}^m = S \cap \mathbb{N}^{j+k}$ admits polynomial cluster sequences.*

Proof. Let $N_1, \ldots, N_m \in \mathbb{N}$ and $p_1, \ldots, p_m \in \mathbb{N}$ be fixed and consider the preimage T of any singleton $(c_1, \ldots, c_m) \in \{0, \ldots, N_1 + p_1 - 1\} \times \ldots \times \{0, \ldots, N_m + p_m - 1\}$ under the corresponding shifted grid projection τ of S. Let $(n_1, \ldots, n_m) \in T$ and let $i \in [m]$ be arbitrary. If the i-th coordinate in T is bounded then we are done. If the i-th coordinate in T is unbounded we distinguish two cases.

In case $1 \leq i \leq j$, we choose the T' to be the part of S spanned by choosing the values of the free variables as $\{(n_1, \ldots, n_{i-1}, n_i + \kappa \cdot pd, n_{i+1}, \ldots, n_j) \mid \kappa \in \mathbb{N}\}$ where p is the lowest common multiple of $p_1, \ldots, p_m$ and d is the common denominator of all of the $\mu_1, \ldots, \mu_k$. Clearly the i-th coordinate of T' is unbounded and T' is infinite. Further, since all other free variables are constant, each dependent variable is a monotone univariate polynomial in the i-th coordinate and thus the sum of all coordinates is bounded by some polynomial in the i-th coordinate. Remains to show $T' \subseteq T$. For this let $(n'_1, \ldots, n'_m) \in T'$. Clearly $n'_\ell \operatorname{srem}_{N_\ell} p_\ell = n_\ell \operatorname{srem}_{N_\ell} p_\ell$ for $\ell \in [j] \setminus \{i\}$, since $n'_\ell = n_\ell$. Additionally $n'_i \operatorname{srem}_{N_i} p_i = n_i \operatorname{srem}_{N_i} p_i$, since $n'_i = n_i + \kappa \cdot pd$ and $n_i \geq N_i$. Furthermore $n'_\ell \operatorname{srem}_{N_\ell} p_\ell = n_\ell \operatorname{srem}_{N_\ell} p_\ell$ for $\ell \in [m] \setminus [j]$, because either $n'_\ell = n_\ell$ or the ℓ-th coordinate is unbounded and thus $n'_\ell \geq n_\ell \geq N_\ell$ and $n'_\ell \equiv_{p_\ell} n_\ell$ since n'_ℓ is dependent on n'_i as a univariate polynomials and for any univariate polynomial $q(x)$ we have $q(x) \equiv_{p_\ell} q(x + \delta)$ if δ is a multiple of $p_\ell \cdot d$. Thus $(n'_1, \ldots, n'_m) \in T$.

Remains to consider the case where $i > j$. Since the i-th coordinate in T is unbounded, n_i is a function depending on at least one of the free variables $n_{i'}$. We repeat the previous argument by continuing with i' instead of i which finishes the proof, since $n_i \geq \alpha \cdot n_{i'}$ for some $\alpha \in \mathbb{R}^+$. □

All in all this combines to the following theorem which characterizes the special case of multivariate polynomial functional promise closure properties.

4.41 Theorem. *Let $S \subseteq \mathbb{Q}^m = \mathbb{Q}^{j+k}$ be a monotone graph variety and let $I = I(S)$ be its vanishing ideal. A multivariate polynomial $\varphi : S \to \mathbb{N}$ is a functional promise closure property of #FA with regard to S if and only if there exists a $\psi \in I$ such that $\varphi + \psi$ is a multivariate functional closure property of #FA.*

Proof. If there exists a $\psi \in I$ such that $\varphi+\psi$ is a multivariate functional closure property of #FA, then $(\varphi+\psi)_{|S} = \varphi_{|S}$ and thus φ is a functional promise closure property of #FA with regard to S by definition. On the other hand if φ is a functional promise closure property of #FA with regard to S, then there is a functional closure property φ' of #FA with $\varphi_{|S} = \varphi'_{|S}$. Note that a priori φ' might not be a polynomial, but just a finite sum of finite products of univariate ultimately PORC functions.

W.l.o.g. let all thresholds N and quasiperiods p of φ' coincide. Let $S' \subseteq \mathbb{N}^j$ be the projection of S to the free variables, i.e. all points $\mathbf{s} \in \mathbb{N}^j$ with $\mu_i(\mathbf{s}) \in \mathbb{N}$ for all $i \in [k]$. Our first goal is now to find an infinite full-dimensional grid $\boxplus'$ within S', that, after adding back the dependent variables using the μ_i, ends up constant under the shifted grid projection τ.

Due to the monotonicity of the μ_i, choosing larger and larger points $\mathbf{s} \in S'$, eventually every $\mu_i(\mathbf{s})$ will either be always $\geq N$ or be constant. Let $\mathbf{o} = (o_1, \ldots, o_j) \in S'$ be a point where this behavior has stabilized and where all $o_i \geq N$. Moreover, let d be the common denominator of $\mu_1, \ldots, \mu_k$. Then

$$\tau(\mathbf{o}, \mu_1(\mathbf{o}), \ldots, \mu_k(\mathbf{o})) = \tau(\mathbf{o} + pd\mathbf{a}, \mu_1(\mathbf{o} + pd\mathbf{a}), \ldots, \mu_k(\mathbf{o} + pd\mathbf{a}))$$

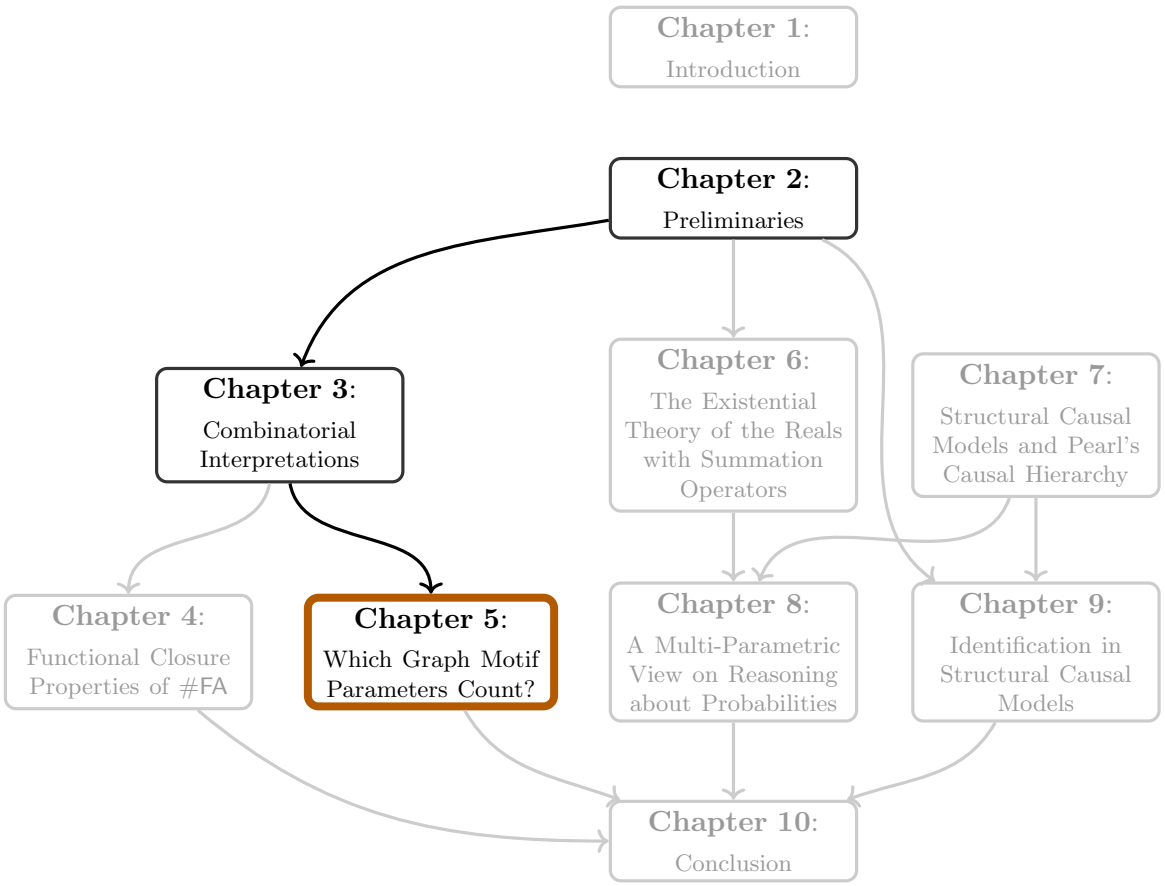
for all $\mathbf{a} \in \mathbb{N}^j$, i.e. we have found our infinite full-dimensional grid $\boxplus' = \{\mathbf{o} + pd\mathbf{a} \mid \mathbf{a} \in \mathbb{N}^j\} \subseteq S'$. Let $\boxplus \subseteq S$ be the lifting of $\boxplus'$ to $\mathbb{N}^m$ by adding in the dependent variables using the μ_i. Since τ is constant for $\boxplus$, the constituents of φ' corresponding to $\boxplus$ form a multivariate polynomial $\varphi^{\boxplus}$ with $\varphi^{\boxplus}_{|\boxplus} = \varphi'_{|\boxplus} = \varphi_{|\boxplus}$. Due to multivariate polynomial interpolation on $\boxplus$[11] we also have $\varphi^{\boxplus}_{|S} = \varphi_{|S}$. Their difference $\varphi^{\boxplus} - \varphi$ is the desired polynomial $\psi \in I$. It remains to show that $\varphi^{\boxplus}$ itself is a multivariate closure property of #FA. Repeating the same argument as in Lemma 4.31 proves that the dominating terms of $\varphi^{\boxplus}$ have positive coefficients, even after replacing some variables by constants. Using Lemma 4.31 again, shows that $\varphi^{\boxplus}$ is a multivariate closure property of #FA.

□

[11] Both φ and $\varphi^{\boxplus}$ can be interpreted as polynomials in the j free variables and $\boxplus$ is a grid on the j free variables, allowing us to use multivariate polynomial interpolation.

4.5 Conclusion

We characterized the functional closure properties of #FA to be precisely the ultimately PORC functions in the univariate case and combinations of ultimately PORC functions in the multivariate case. Additionally we characterize promise functional closure properties of #FA with regard to some natural families of sets S. Natural further directions of research are now whether we can characterize the promise functional closure properties of #FA for more sets S and whether our methods can be applied to characterize functional closure properties for more powerful models of computation.



Which Graph Motif Parameters Count?

5.1 Introduction

5.1.1 Do Nonnegative Graph Motif Parameters Count?

A graph F is an *induced subgraph* of G if F can be obtained from G by deleting vertices. An *induced copy* of a graph H in G is an induced subgraph F of G that is isomorphic to H. For two graphs H and G, write $\#\mathrm{Ind}(H \to G)$ for the number of induced copies of H in G, and when H is fixed, write $\#\mathrm{Ind}(H \to \star)$ for the function mapping $G \mapsto \#\mathrm{Ind}(H \to G)$. As defined by Curticapean, Dell, and Marx [CDM17], a *graph motif parameter* is a rational linear combination of such functions; see Section 5.2.3 for formal definitions. The step from individual counts to linear combinations turns graph motif parameters into a rich function space that captures interesting counting problems related to small patterns. For example, the number of (not necessarily induced) subgraphs isomorphic to a fixed k-vertex graph H is a nonnegative linear combination of induced subgraph counts from all k-vertex supergraphs $H' \supseteq H$. This means that (not necessarily induced) subgraph counts from fixed graphs are graph motif parameters. The same holds for the number of homomorphisms from a fixed graph H. Moreover, the coefficients of a graph motif parameter in its linear combination over $\#\mathrm{Ind}(H \to \star)$ are unique.

[1]This holds because $\#K_1{}^2$ counts the pairs $(u, v) \in V(G)^2$, and so does the linear combination $2\#K_2 + 2\#I_2 + \#K_1$, whose summands correspond to the isomorphism type of $G[\{u, v\}]$, which is either K_2, I_2, or K_1. As we will elaborate later, graph motif parameters enjoy a general linearization property: Any polynomial combination of induced subgraph counts can be expressed as a unique linear combination. This holds even for "subobject" counts under more general notions of objects; see Lemma 5.44 and Corollary 5.45. This motivates our focus on *linear* combinations.

Graph motif parameters have been studied thoroughly before, mostly from a complexity-theoretic view [JM15b; JM15a; JM17; CDM17; Rot19; RSW20; RSW21b; DRSW22; DMW24]: Each fixed graph motif parameter f can be evaluated in polynomial time on n-vertex input graphs, as it involves a linear combination of induced subgraph counts from fixed graphs. Under complexity-theoretic assumptions, the above works give lower bounds on the exponent of this polynomial. We focus on the separate question which graph motif parameters have local combinatorial interpretations.

For example, most of the above papers consider unweighted sums (i.e., linear combinations with coefficients 0 or 1) of induced subgraph counts from a set X of k-vertex graphs. This has the evident local combinatorial interpretation of counting k-vertex subsets S of a graph G that induce a graph $G[S]$ from X. More generally, we can allow arbitrary integers as coefficients; as we show in Lemma 5.12, all *integer-valued* graph motif parameters f can be obtained this way. In this setting, it is possible for f to have negative integer coefficients but still satisfy $f(G) \geq 0$ on all graphs G.

5.1 Example. Consider the quantity $(|V(G)| - 1)^2 = (n-1)^2 \geq 0$ for n-vertex graphs G. This is nonnegative and, in fact, a graph motif parameter: Abbreviating $\#H = \#\mathrm{Ind}(H \to G)$, we have $n = \#K_1$ and $1 = \#K_0$, and $(n-1)^2 = \#K_1{}^2 - 2\#K_1 + \#K_0$. Moreover, the quadratic term $\#K_1{}^2$ can be linearized[1] , i.e., expressed as the linear combination $\#K_1{}^2 = 2\#K_2 + 2\#I_2 + \#K_1$. Thus, we have

$$0 \leq (\#K_1 - 1)^2 = 2\,\#K_2 + 2\,\#I_2 - \#K_1 + \#K_0,$$

and have constructed a linear combination of induced subgraph counts evaluating to a nonnegative value for every graph G, even though the term $\#\mathrm{Ind}(K_1 \to \star)$ appears with coefficient -1.

This example in fact has a very simple local combinatorial interpretation for graphs G with $n \geq 1$ vertices: After fixing an arbitrary vertex z_0, the quantity $(n-1)^2$ simply counts the pairs $(u, v) \in V(G)^2$ with $u \neq z_0$ and $v \neq z_0$. Let us consider a example where finding such an interpretation seems trickier:

5.2 Example. Consider the graph motif parameter

$$f(G) \;=\; \#\,\mathbf{I} - \#\,\mathbf{V} + \#\,\mathbf{V}\!-\!\bullet + 2\,\#\,\mathbf{V}\!\!\bullet + 4\,\#\,\mathbf{V}\!\!\bullet \;\geq\; 0\,.$$

We postpone the proof of its non-negativity to Section 5.9 for the interested reader. Contrary to our previous example, the graphs in this linear combination feature no isolated vertices. Does f have a local combinatorial interpretation?

Our main result in Theorem 5.14 gives a formal argument against f having a local combinatorial interpretation: In a nutshell, a nonnegative graph motif parameter involving only patterns without isolated vertices has a local combinatorial interpretation if and only if all its coefficients are nonnegative. (The "if" part of this result is trivial.) In the next section, we will give the relevant formal definitions to state our results in Section 5.3 and compare them to related work in Section 5.4.

5.2 Preliminaries

5.2.1 Type-2 Counting Complexity

We study computational problems on inputs given by concise descriptions. Such problems have been studied by Johnson, Papadimitrou, and Yanakakis [JPY88; Pap94], who introduced subclasses of the complexity class TFNP, such as PLS or PPAD, to capture search problems in exponentially large objects. The problems in these subclasses of TFNP all have a similar structure: Each input x implicitly encodes an exponentially large structure, e.g., a graph G via a Boolean circuit x that computes the edge relation of the graph G. The witness sought in the search problem is however still polynomially large in $|x|$, which is tiny compared to the size of the encoded input instance. The guiding examples in our paper are induced subgraphs H of a fixed size in an exponentially large graph G. In this problem, the $O(1)$ labels of the vertices in G that induce H constitute a witness.

A natural abstraction of this setting is obtained by providing the input as an oracle, rather than via a succinct encoding, see e.g. [BIP95]. In this setting, which is denoted as *type-2 complexity*, we can only query the given structure, for instance by making queries to the edge relation of the graph, but we do not have access to a circuit computing the

edge relation. Type-2 problems also arise naturally in computability and complexity studies of real functions, see e.g. [Wei00; Ko91]. Since a real number is an infinite object, it is given as an oracle which one can query to get better and better rational approximations.

To define the type-2 framework formally, fix some alphabet Σ. We assume that $0, 1 \in \Sigma$, which are interpreted as false and true, respectively. An *oracle* is a total function of type $\Sigma^* \to \{0,1\}$. The computational model in the type-2 framework is an oracle Turing machine, which gets an input string x together with access to an oracle O.

5.3 Example. For a fixed graph H, we consider the problem of finding an induced H-copy in a large graph G. The graph G is given as (x, O) where $O : \Sigma^* \to \{0,1\}$ is an oracle encoding the edge relation and x is the number of nodes of G written in binary. The nodes of G are $0, \dots, x-1$, written as bit strings of fixed length n. Given two nodes u and v of G, there is an edge between u and v iff $O(uv) = 1$. (Since we consider strings of fixed length n, no separation symbol is needed.) In this representation, most of the information about G is stored in the oracle; the input x only encodes the size of the graph, and the running time of oracle Turing machines is measured in terms of x.

A consistency issue can arise in this definition: Since the graph G is exponentially large, the oracle Turing machine might not be able to check whether the encoding is consistent. For instance, one might wish to encode an undirected graph G but have $O(uv) \neq O(vu)$. In the case of graphs, such inconsistencies can be "repaired" by avoiding observing them, e.g., by only querying strings uv with $u < v$. In the context of TFNP and its subclasses, such issues are often elegantly repaired by interpreting any oracle as a graph with the desired properties. In general however, such a fix may not be possible for more complicated structures, in particular in the abstract setting of category theory that we will also study. Therefore, we will also resort to promise problems, in which a Turing machine is only required to work properly on inputs (x, O) that are a proper encoding of an input object; see Definition 5.8 for more details. Note that this also encompasses the previous solutions. If we can repair the oracle or choose an interpretation such that every oracle encodes a valid instance, then we simply choose the set of "don't care" instances to be the empty set.

A k-ary *type-2 relation* R gets a tuple of k strings x as an input, has access to an oracle O, and outputs a value $R(x, O) \in \{0,1\}$. Typically, we will consider unary type-2 relations, but relations with higher arity naturally occur when one considers witnesses, see below, when we define the existential and counting operators. A type-2 relation R is polynomial-time computable if there is an oracle Turing machine M that given input x and oracle access to O computes $R(x, O)$ in polynomial-time in the length of the input x. Oracle queries have unit costs, but M has to write the query strings on the oracle tape. Note that all time bounds we consider depend only on the length $|x|$ of x, but not on x itself and, more importantly, not on the oracle.

If there is a polynomial-time computable $(k+1)$-ary type-2 relation S, we define the k-ary type-2 relation $\exists S$ by $\exists S(x, O) = 1$ iff there is a polynomially long y (in the length $|x|$) such that $S(x, y, O) = 1$.

5.4 Example. In our example of finding an induced copy of H in the graph G, we start with a binary type-2 relation $\mathrm{IsIndSub}_H(x, y, O)$. The relation interprets y as an encoding of a set Y of k distinct nodes of G with k being the number of nodes of H. The relation then queries the oracle O to obtain the induced subgraph $G[Y]$. It then checks by brute force whether $G[Y]$ and H are isomorphic. If yes, it returns the value 1 and otherwise 0. This is clearly polynomial-time computable in $|x|$. The relation $\exists\mathrm{IsIndSub}_H(x, O)$ now returns 1 if G contains an induced copy of H *somewhere*, and 0 if G does not contain such a copy. See also Lemma 5.13 for a more formal proof.

As in classical type-1 complexity, we can also introduce a counting operator # to type-2 complexity, similar to the existential operator $\exists$: A polynomial-time computable $(k+1)$-ary type-2 relation S defines a k-ary type-2 counting problem by $\#S(x, O) = \#\{y \mid S(x, y, O) = 1\}$, where y is polynomially long.

5.5 Example. Continuing the example above, $\#\mathrm{IsIndSub}_H$ now counts the number of induced subgraphs in G that are isomorphic to H, that is, $\#\mathrm{IsIndSub}_H(x, O) = \#\mathrm{Ind}(H \to G)$, where (x, O) is the oracle encoding of G.

The goal of this paper is to characterize those integer valued nonnegative graph motif parameters that have a local combinatorial interpretation. In other words, we ask which nonnegative graph motif parameters count something. The informal notion of "local combinatorial interpretation" is formalized below as follows:

5.6 Definition. $\#\mathsf{P}^{\circlearrowleft} = \{\#R \mid R \text{ is a polytime computable binary type-2 relation}\}$.

In the notation $\#\mathsf{P}^{\circlearrowleft}$, the dotted circle represents a placeholder for the input oracle.

5.7 Definition. For a graph motif parameter φ, we denote by Eval_φ the corresponding type-2 function, which gets the graph G given in the form (x, O). An integer valued nonnegative graph motif parameter φ is called *locally combinatorially interpretable* if $\mathrm{Eval}_\varphi \in \#\mathsf{P}^{\circlearrowleft}$ (depending on context also $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$, see later for an explanation).

It is open for discussion whether $\#\mathsf{P}^{\circlearrowleft}$ exactly captures "nonnegative combinatorial interpretation" in the informal sense used by combinatorists. If a problem is however *not* in $\#\mathsf{P}^{\circlearrowleft}$, then it at the very least requires some global operations. Our dichotomy result for integer-valued nonnegative graph motif parameters is particularly clean-cut in this regard: Such a parameter is in $\#\mathsf{P}^{\circlearrowleft}$ iff all its coefficients are nonnegative integers, so it indeed counts induced subgraphs.

To study more complicated structures, we consider promise problems, since the oracle TM might not be able to check whether an oracle is a legal encoding. Hence, we will have a set of "don't care" inputs (x, O), which will correspond to illegal encodings of

input objects in our setting. Our oracle TM only needs to compute the correct result on inputs which are not "don't care" inputs. One such example are vector spaces, where we interpret the oracle as the characteristic function of the given vector space and it is not immediate how to check whether the oracle indeed describes a vector space. In this case our oracle Turing machine only has to work properly if the given oracle indeed encodes a vector space.

5.8 Definition. A counting function F together with a set D of "don't care" inputs is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot}$ if there is a polynomial-time computable binary type-2 relation R such that $\#R$ and F coincide on all inputs (except potentially D).

Every function in $\#\mathsf{P}^{\cdot\cdot}$ is also in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot}$ with the empty set of "don't care" inputs.

5.2.2 What is a Local Combinatorial Interpretation—and What is Not?

Intuitively, a local combinatorial interpretation should mean "counting something". But that is not enough. Even the graph motif parameter f from Example 5.2 counts something: For a given graph G, we consider the set of witnesses for each pattern in f. So $W($$)$ are all (induced) occurrences of an edge in G, $W($$)$ are all (induced) occurrences of a triangle in G and so on. The sets $W($$)$ and $W($$)$ will be multi-sets containing two and four copies of the witnesses, respectively, due to the coefficients. Let W_+ be the union of all witness sets that correspond to a term with positive coefficient and $W_- = W($$)$ be all witnesses that correspond to a term with negative coefficient. Since f is nonnegative, $|W_+| \geq |W_-|$. Thus there is an injective map $i : W_- \to W_+$. So we could say that f counts all elements in W_+ that are not in $\mathrm{im}(i)$. However, this "cheating away" of the difference $|W_+| - |W_-|$ is unsatisfactory, because it is (computationally) hard to decide whether a given element in W_+ should be counted or not.

Counting complexity gives a way to define local combinatorial interpretations: We are given a set of small witnesses $W \subseteq \{0,1\}^*$, encoded as binary strings, and for each $x \in \{0,1\}^*$ we can decide in polynomial time whether it is in W or not. This rules out the construction above, since constructing the embedding i is infeasible. Computing the size $|W|$ is a problem in the complexity class $\#\mathsf{P}$: A nondeterministic Turing machine guesses a witness x and then deterministically verifies whether it is in W. The number of accepting paths is precisely $|W|$.

In the case of graph motif parameters, the witnesses are given as ordered lists of nodes having constant length, say k, since only the target graph G varies. If G has N nodes, each node can be encoded by a bitstring of length $\log N$ and the total witness size is $k \log N$. When the running time should be polynomial, then the Turing machine M for checking the witness needs random access to the graph, because it cannot inspect the whole graph. This is modeled using oracles and type-2 complexity.

5.2.3 Induced Subgraph Numbers

Graphs in this paper are finite, undirected, and simple, i.e., they do not contain multi-edges or self-loops. Given a graph G, we write $V(G)$ and $E(G)$ for its vertex and edge set, respectively. We write $\mathcal{G}$ for the class of all graphs. Towards applying a Ramsey theorem, we will also consider *ordered* graphs: These are graphs $G = (V, E, \leq_G)$ where (V, E) is a graph and $\leq_G$ is a total order on V. We write $\mathcal{OG}$ for the class of these graphs. When ordered graphs are represented by an oracle, we always assume the natural lexicographic order on the vertices.

Given graphs H and G, we say that H is a *subgraph* $H \subseteq G$ of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. In the case of ordered graphs $H = (V, E, \leq_H)$ and $G = (V', E', \leq_G)$, we additionally require that $\leq_H$ is the restriction of $\leq_G$ to V. Given a vertex set $X \subseteq V(G)$, write $G[X] = (X, \{vw \in E(G) \mid v, w \in X\})$ for the *subgraph* of G *induced* by X. We write $H \sqsubseteq G$ to denote that H is an induced subgraph of G, and we write $\mathcal{P}(G)$ for the poset of induced subgraphs of a graph G, ordered by $\sqsubseteq$. We call H *pure* if it contains no isolated vertices, i.e., no connected components isomorphic to K_1, and we write $\mathcal{G}^{\text{pure}}$ for the class of pure graphs and $\mathcal{OG}^{\text{pure}}$ for the pure ordered graphs.

A *homomorphism* from a graph H into a graph G is a function $f : V(H) \to V(G)$ such that $uv \in E(H)$ implies $f(u)f(v) \in E(G)$; we write $\#\text{Hom}(H \to G)$ for their number. A homomorphism is an *embedding* if it is injective; we write $\#\text{InjHom}(H \to G)$ for their number. A *strong embedding* is an embedding with the additional condition that $uv \notin E(H)$ also implies $f(u)f(v) \notin E(G)$; we write $\#\text{StrInjHom}(H \to G)$ for their number. An *isomorphism* is a strong embedding that is bijective; two graphs H and G are isomorphic if there is an isomorphism from H to G. For isomorphic graphs H, G we write $H \cong G$. All of these definitions apply to ordered graphs as well; in this setting, homomorphisms f must additionally preserve the orderings of H and G, i.e., if $u \leq_H v$, then $f(u) \leq_G f(v)$.

An automorphism of H is a strong embedding from H into H; we write $\#\text{Aut}(H)$ for the numbers of automorphisms of H. The numbers of embeddings and strong embeddings from a graph H into some graph G are multiples of $\#\text{Aut}(H)$. We obtain the number of subgraphs isomorphic to H as

$$\#\text{Sub}(H \to G) = \#\text{InjHom}(H \to G)/\#\text{Aut}(H)$$

and the number of induced subgraphs isomorphic to H as

$$\#\text{Ind}(H \to G) = \#\text{StrInjHom}(H \to G)/\#\text{Aut}(H)\,.$$

Note that for ordered graphs H, we have $\#\text{Aut}(H) = 1$. For a graph H, we write $\#\text{Ind}(H \to \star)$ for the function that maps $G \mapsto \#\text{Ind}(H \to G)$. We use that these functions are linearly independent for non-isomorphic graphs H, even when restricted to particular domains:

5.9 Fact. Let $\mathcal{G}' \subseteq \mathcal{G}$ be a finite set of pairwise non-isomorphic graphs. For $H \in \mathcal{G}'$, write $f_H : \mathcal{G}' \to \mathbb{Q}$ for the restriction of $\#\mathrm{Ind}(H \to \star)$ to $\mathcal{G}'$. Then the functions $\{f_H \mid H \in \mathcal{G}'\}$ are linearly independent. The same applies for ordered graphs.

Proof. We show the fact for unordered graphs; the version for ordered graphs is shown along the same lines. Let $\{G_1, \ldots, G_t\}$ for $t \in \mathbb{N}$ be an enumeration of $\mathcal{G}'$ such that $|V(G_i)| < |V(G_j)|$ implies $i < j$. Consider the matrix A whose rows and columns are indexed by $G_1, \ldots, G_t$, with entries $\#\mathrm{Ind}(G_i \to G_j)$.

We show that this matrix is lower triangular with all diagonal entries equal to 1, implying full rank and thus linear independence of the functions f_H: Consider indices i, j with $i \geq j$: First, we have $|V(G_i)| \geq |V(G_j)|$ by the choice of enumeration. If $A(i,j) \neq 0$ holds, it must follow that $|V(G_i)| = |V(G_j)|$, as otherwise G_i has too many vertices to be an induced subgraph of G_j. But then $A(i,j) \neq 0$ only for $j = i$. In this case, $A(i,j) = 1$. □

Having established that the functions $\#\mathrm{Ind}(H \to \star)$ are linearly independent, we consider linear combinations of these basis functions. The following term was coined in [CDM17]:

5.10 Definition. A *graph motif parameter* is a function $\varphi : \mathcal{G} \to \mathbb{Q}$ that admits pairwise non-isomorphic pattern graphs $H_1, \ldots, H_s$ and coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$ such that, for all graphs G,

$$\varphi(G) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(H_i \to G). \tag{5.11}$$

We say that φ is *pure* if every graph H_i with non-zero coefficient is pure. An *ordered graph motif parameter* similarly is a linear combination of induced subgraph counts from pairwise non-isomorphic *ordered* graphs. The *support* $\mathrm{supp}(\varphi)$ is the set of all H_i, s.t. $\alpha_i \neq 0$.

Note that φ being pure is well-defined, as the linear combination (5.11) is unique by Fact 5.9: The (isomorphism types of) graphs and coefficients appearing in the linear combination are uniquely determined; we therefore speak of *the coefficients* and *the patterns* of φ. Since we investigate whether or not a graph motif parameter φ has a local combinatorial description, i.e., whether φ counts a set of objects, we will restrict ourselves to graph motif parameters with image $\mathbb{N}$, as all others can clearly not count anything. By the following lemma, this implies that the coefficients in the linear combination (5.11) can be assumed to be integers.

5.12 Lemma. *Let φ be an (ordered) graph motif parameter. Then $\varphi(G) \in \mathbb{Z}$ for every (ordered) graph G if and only if all coefficients of φ are integers.*

Proof. The "if" is obvious, since $\#\mathrm{Ind}(H \to G) \in \mathbb{Z}$ for any H, G. For the "only if", let H be a pattern in φ with a coefficient $\alpha_H \in \mathbb{Q} \setminus \mathbb{Z}$, such that $|V(H)|$ is minimal among all such patterns. Then $\#\mathrm{Ind}(H \to H) = 1$ and $\varphi(H) = \alpha_H + \beta \notin \mathbb{Z}$ for some $\beta \in \mathbb{Z}$, since all other basis functions with non-integer rational coefficients evaluate to 0 on H, as their pattern has at least $|V(H)|$ vertices and is not isomorphic to H. □

5.13 Lemma. *Let* $\varphi(G) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(H_i \to G)$ *with all* $\alpha_i \in \mathbb{N}$. *Then* $\mathrm{Eval}_\varphi \in \#\mathsf{P}^{\circlearrowleft}$. *This holds for unordered and ordered graph motif parameters.*

Proof. Since φ is a positive integer linear combination of finitely many induced subgraph numbers, and $\#\mathsf{P}^{\circlearrowleft}$ is closed under such linear combinations, it suffices to consider $\varphi(G) = \#\mathrm{Ind}(H \to G)$ for $H \in \mathcal{G}$. We show the statement for unordered graphs, the proof for ordered graphs is analogous.

A nondeterministic Turing machine for Eval_φ proceeds as follows, for hardcoded H with $|V(H)| = k$: Given (x, O) encoding a graph G, nondeterministically guess strings $x_1 < \ldots < x_k < x$ of length $|x|$, where $<$ denotes the lexicographic ordering on strings. Then, for every $\{i, j\} \in \binom{k}{2}$, write $x_i x_j$ on the oracle tape and query O to obtain a bit $a_{i,j}$. Finally, test by brute-force whether the graph with adjacency matrix given by $a_{i,j}$ is isomorphic to H; this only requires constant time, as H is fixed. If the test succeeds, then this branch accepts, otherwise it rejects. This nondeterministic Turing machine clearly runs in polynomial-time on every branch, and the number of accepting computation paths equals the number of vertex-subsets S of G such that $G[S]$ is isomorphic to H. □

5.3 Our Results

5.3.1 Graphs

We characterize the pure graph motif parameters φ with a local combinatorial interpretation as those whose coefficients are nonnegative integers. Such graph motif parameters φ clearly have a local combinatorial interpretation, as they count all induced occurrences of pattern graphs, possibly with multiplicities (see Lemma 5.13.) On the other hand, if φ has a negative coefficient, then we show that φ is not in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ and thus does not admit a local combinatorial interpretation. Formally we prove (recall Def. 5.7 about local combinatorial interpretability):

5.14 Theorem. *Let* φ *be a pure graph motif parameter, i.e., we have* $\varphi(G) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(H_i \to G)$ *for pairwise non-isomorphic pure* $H_1, \ldots, H_s$ *and coefficients* $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$.[2] *Then* φ *is locally combinatorially interpretable iff all* $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$ *are nonnegative integers.*

[2] By Lemma 5.12 the interesting cases are those where all $\alpha_i \in \mathbb{Z}$.

We show Theorem 5.14 in Section 5.5 with a delicate and technically demanding proof. The intuition is however rather easy to explain, and we illustrate it on the Example 5.2. An NTM M is supposed to output $f(G)$, but it can only query a small part of G. Assume that G is either the union of a triangle and exponentially many singletons, or the union of an edge with exponentially many singletons, or G is the graph that consists of only exponentially many singletons. If G is an edge and singletons, then one computation path of M must accept and must have queried the oracle at the edge, because otherwise this computation path would lead to an accepting path even if G has no edges, which would be the wrong answer. Now, in the case that G is a triangle and singletons, it turns out (which is delicate) that such an accepting path exists for each of the triangle edges, so we get 3 accepting computation paths, but $f(G) = 2$, so the NTM has too many accepting paths.

5.3.2 More General Structures

The methodology used to prove Theorem 5.14 can be generalized to more general structures, provided that they satisfy the following conditions:

- Ramsey property: We require Ramsey-type theorems for the structures, which assert—roughly speaking—that for given objects A, B, there exists an object C which contains enough B-copies such that, by partitioning the set of all A-copies in C into a small number of parts, there is some part that contains a full B-copy. In the case of pure graph motif parameters, for instance, this is Lemma 5.22: For any ordered graphs F, G, there is an ordered graph H such that for any partitioning of the induced subgraphs of H that are isomorphic to F, we can find an induced copy of G in H such that all induced copies of F in this copy of G are in the same part.
- A neutral padding operation: We require an operation to enlarge a given object without increasing the number of induced subobjects. In the case of pure graph motif parameters, this can be achieved by adding isolated vertices, however we introduce different possible ways of achieving this for different objects, leading to different definitions of "pure".

Relational Structures

For instance, we can prove results similar to Theorem 5.14 for various variants of relational structures, which generalize graphs. In graphs, we consider exactly one binary relation on a set V, the edge relation. In a relational structure, we can have multiple relations of various arities. A relational structure A is an induced substructure of another relational structure B, if we can obtain A by restricting the relations of S to the vertices of A. Similarly to graphs, we can now define *motif parameters* as linear combinations of

induced substructure numbers. Formal definitions are given in Section 5.7. We get the following classification for motif parameters of relational structures.

5.15 Theorem (informal, see Theorem 5.31 for a precise statement)**.** *The evaluation function of a pure motif parameter of relational structures is locally combinatorially interpretable (i.e., it is in* $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$*) iff all its coefficients are nonnegative integers.*

We also consider further types of relations, namely multiset relations (each relation is a multiset and can have repetitions), and list relations (which is the equivalent to directed graphs in the relational setting) with and without repetitions. We call these relational structures *mixed*. Precise definitions are again given in Section 5.7. For such mixed relational structures, we get similar dichotomies, which we first prove in the ordered setting (Theorem 5.36), since the corresponding Ramsey property only holds for ordered mixed relational structures. Then we transfer this theorem to the unordered setting in Theorem 5.37—a similar detour into the ordered setting is in fact already needed for the case of graphs.

An application of mixed relational structures are colored graphs, where nodes are colored and isomorphisms have to respect the colors. Theorem 5.38 shows a dichotomy result for colored graphs.

Category Theory

Ramsey-type theorems have even been established within category theory [GLR72; NR77]. Since Ramsey arguments are crucial in our method, this raises the interesting question whether we can also transfer our results into this realm. Indeed, we achieve this in Section 5.8. Our objects of consideration are now objects of some category C, and the subobjects we consider are now specified using a class of morphisms $\mathcal{M}$ from C. For two objects a and b of C, we call the morphisms $a \to b$ in $\mathcal{M}$, identified if they only differ by an isomorphism of a, the $\mathcal{M}$*-subobjects* of a under b. In the case of undirected graphs, we choose $\mathcal{M}$ to be the class of all strong graph embeddings. An important concept in our construction will be so-called factorization systems, used also by Lagodzinski [Lag24] and Isbell [Isb91] in the context of counting problems. We have a second class of morphisms $\mathcal{E}$ and we require that every morphism $f : a \to b$ factors (essentially uniquely) as $f = me$ with $m \in \mathcal{M}$ and $e \in \mathcal{E}$. In our running example of unordered graphs, we can take $\mathcal{E}$ to be the class of surjective graph homomorphisms and $\mathcal{M}$ the above class of strong graph embeddings. Then every graph homomorphism $f : H \to G$ factors through the subgraph $G[f(V(H))]$ of G that is induced by the image of $V(H)$.

Recall that for graph motif parameters, we had a notion of pure graphs. We will have the same here: We will have two categories, a category C of objects that we consider and a category P of pure objects that we want to count. In Lemma 5.54, we collect a number of natural and rather mild requirements on C and P (maybe except for the property of

being Ramsey) in order to prove our main result that P-pure motif parameters can only have a local combinatorial interpretation if the coefficients are all nonnegative.

5.16 Theorem (informal, see Theorem 5.58 for the precise statement)**.** *Let φ be a P-pure motif parameter. Then under the conditions specified in Lemma 5.54, we have* $\mathrm{Eval}_\varphi \notin \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ *unless all its coefficients are nonnegative integers.*

The theorem does not automatically give a dichotomy, since we do not have a corresponding upper bound when the coefficients are all nonnegative integers. However, our concrete applications in fact do admit corresponding upper bounds:

- First we consider finite dimensional vector spaces over finite fields, which were shown to have the Ramsey property in [GLR72] using methods from category theory. In particular, we show that the evaluation function of a motif parameter over a finite vector space is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ iff all its coefficients are nonnegative integers (Theorem 5.60).
- Then we consider so-called parameter sets, which are subsets of A^n for some fixed alphabet A where we may set a coordinate to constant or identify any number of coordinates. They are also known to satisfy a Ramsey property, see [Neš95, Theorem 10.4]. As a consequence of our general theorem, a motif parameter over parameter sets is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ iff all its coefficients are nonnegative integers (Theorem 5.62).

5.3.3 Linearization

As shown examplarily on the first pages, graph motif parameters enjoy a useful linearization property: Any polynomial in induced subgraph counts is a graph motif parameter, that is, a *linear* combination of induced subgraph numbers. Linearization motivates our focus on linear combinations of subobject counts.

In Lemma 5.44 and Corollary 5.45, we establish conditions for subobject numbers of a category C to enjoy similar linearization properties. We obtain that the coefficients in the linearized version are positive and have a local combinatorial interpretation if the polynomial we started with has positive coefficients in the binomial basis; see the next section.

5.4 Related Work

Searching for combinatorial interpretations for nonnegative integer quantities has a long tradition in combinatorics dating back to Cayley, Littlewood, Richardson, Schützenberger, Macdonald, Schensted, Knuth, Stanley, and others, e.g., [Kos82; Sch77; Sta00], and has recently been studied from the perspective of $\#\mathsf{P}$ containment and functional $\#\mathsf{P}$ closure properties [Pak19; IP22; IPP23; CP24]. The univariate relativizing functional closure properties of $\#\mathsf{P}$ (and thus the functional closure properties of $\#\mathsf{P}^{\cdot\cdot\cdot}$) have been

characterized in [Cai+89, Thm 3.1.1(b)]: Consider the binomial coefficient $\binom{x}{i} = \frac{1}{i!}x(x-1)(x-2)\cdots(x-i+1)$ for $i \in \mathbb{N}$ as a polynomial in x, and let $\varphi : \mathbb{N} \to \mathbb{N}$ be a univariate polynomial with expansion $\varphi(x) = \sum_{i=1}^{k} \alpha_i \binom{x}{i}$ for $\alpha_i \in \mathbb{Q}$. This expansion is called the binomial basis expansion of φ. It is known that $\varphi(\#\mathsf{P}^{\circlearrowleft}) \subseteq \#\mathsf{P}^{\circlearrowleft}$ iff all $\alpha_i \in \mathbb{N}$ (see [IP22] for a proof).

In the language of graphs, the construction in [Cai+89] can be phrased as follows. Let $\mathcal{G}'$ denote the graphs on $\{1, 2, \ldots, 3k\}$ whose vertices $\{3i, 3i+1, 3i+2\}$ either form a triangle or a path •-• on 3 vertices. Consider graph motif parameters for which all patterns are disjoint unions of triangles, i.e., $\varphi(G) = \sum_i \alpha_i \#\mathrm{Ind}(H_i \to G)$, where H_i is a union of i disjoint triangles. For $G \in \mathcal{G}'$ with x many triangles, we have $\varphi(G) = \sum_i \alpha_i \binom{x}{i}$. Their construction can be interpreted as oracle graphs $G \in \mathcal{G}'$ showing that φ is not locally combinatorial.

The non-oracle implications in the univariate case were studied in [OH93, p. 307], which shows that $\mathsf{GapP} \cap \{f : \{0,1\}^* \to \mathbb{N} \mid \forall w : f(w) \geq 0\} = \#\mathsf{P}$ implies $\mathsf{C_{=}P} = \mathsf{co\text{-}NP}$ and $\mathsf{SPP} = \mathsf{UP}$.[3] The multivariate case was resolved in [HVW95, Thm 3.14], cf. [Her95, Thm 23], based on a Ramsey theorem [Her95, Thm 12]. As in the univariate case, one gets small variants of our main theorem, whereas we consider a larger set of non-isomorphic graphs of the same cardinality, each without isolated vertices. The functional closure properties of other counting classes are mainly unknown, with the exception of GapP [Bei97, Thm 6] and finite automata (see Chapter 4).

[3] See [HO02] for an overview over these classes. You will however not need to know about them to understand this work.

While search versions of type-2 problems have been studied already in [Bea+95] and later works on TFNP, the counting versions have only recently been studied to understand whether their nonnegative transformations still have local combinatorial interpretations[4] [IP22]. We improve on the work [IP22] in various ways. First, we show much more general results due to our general theorem in the categorial framework. Second, our results are also valid for unordered objects, like unordered graphs, which do not have the required Ramsey property per se. We overcome this technical difficulty by carefully reducing the unordered case to the ordered one. All graph problems in [IP22] are on ordered graphs only.

[4] They were only called combinatorial interpretations in [IP22], but due to the restriction to type-2 fit into what we call local combinatorial interpretations in this work.

5.5 Results for Graphs

5.5.1 Reduction to Ordered Graphs

To prove Theorem 5.14, we first prove the following result about ordered graphs:

5.17 Theorem. *Let $\varphi : \mathcal{OG} \to \mathbb{N}$ be a pure ordered graph motif parameter. Then $\mathrm{Eval}_\varphi \in \mathsf{Pr\text{-}\#P}^{\circlearrowleft}$ iff all coefficients of φ are nonnegative integers.*

Theorem 5.14 then follows by the simple observation that

$$\#\mathrm{Ind}((V_i, E_i) \to (V, E)) = \sum_{\leq_{V_i}} \#\mathrm{Ind}((V_i, E_i, \leq_{V_i}) \to (V, E, \leq_V)) \tag{5.18}$$

where $\leq_V$ is any arbitrary linear ordering of V and $\leq_{V_i}$ sums over all linear orders of V_i that result in non-isomorphic ordered graphs. Note that all the pattern graphs occurring on the right-hand side are non-isomorphic pure ordered graphs. Even more, for different patterns on the left hand side, all possible occuring patterns on the right hand side are non-isomorphic. Now let $\varphi : \mathcal{G} \to \mathbb{N}$ be a pure graph motif parameter with a negative coefficient. Then the pure ordered graph motif parameter $\varphi' : \mathcal{OG} \to \mathbb{N}$ obtained by applying (5.18) to each of the basis functions has a negative coefficient and thus $\mathrm{Eval}_{\varphi'} \notin \mathsf{Pr\text{-}\#P}^{\cdot\cdot\cdot}$ by Theorem 5.17. Assume for the sake of contradiction that $\mathrm{Eval}_{\varphi} \in \mathsf{Pr\text{-}\#P}^{\cdot\cdot\cdot}$, then we can compute $\mathrm{Eval}_{\varphi'}$ by simulating the unordered oracle in polynomial time (by forgetting the order) and running the corresponding NTM computing Eval_{φ}. This places $\mathrm{Eval}_{\varphi'}$ in $\mathsf{Pr\text{-}\#P}^{\cdot\cdot\cdot}$, a contradiction.

While this only proves that pure graph motif parameters with some negative coefficients are not locally combinatorially interpretable, Lemmas 5.12 and 5.13 complete the proof by proving the reverse direction.

5.5.2 Ordered Graphs

For some subset $D \subseteq \mathcal{OG}$, we say that an ordered graph motif parameter $\Psi : D \to \mathbb{N}$ is *D-good* if all its coefficients are nonnegative integers and all its patterns are from D. If any such coefficient is negative, it is instead called *D-bad*. If the domain D is clear from the context, we will also simply call them *good* and *bad* respectively.

The proof of the hardness in Theorem 5.17 roughly proceeds in two parts:

The first part of the proof finds a target graph G such that, if there is any counterexample where any (restricted) good function would disagree with our bad φ, it must already occur on some induced subgraph of G. For a fixed bad pure ordered graph motif parameter $\varphi : \mathcal{OG} \to \mathbb{N}$ such a counterexample will always exist.

The second part of the proof then turns $\mathcal{P}(G)$ into oracle graphs. We cannot do this directly however, since these instantiations would not all have the same size, allowing the Turing machine to infer (through the number of vertices) possibly useful information about an instantiation without explicitly querying the oracle.

Simply padding all graphs of $\mathcal{P}(G)$ to the same size is still not enough. The main goal here is that acceptance of a computation path is monotone relative to which part of the graph is observed by a computation, i.e., if we embed $H \sqsubseteq G$, then all previously accepting paths that accept on oracle input H should still be accepting when the oracle input is G.

Our proof follows the structure of [IP22]. We start by formalizing computation paths, leading towards the second part of the proof:

5.19 Definition. A *computation path* τ of a nondeterministic Turing machine on some input is defined as the sequence of its nondeterministic choice bits and the answers to its oracle queries (both types of bits appear in the same list, ordered chronologically). Formally, it is an element of $\{0,1\}^*$.

The same Turing machine can yield the same computation path on different inputs, for example, when not the whole input is read, or when having access to different oracles, because the oracles can differ in positions that are not queried.

Given a nondeterministic Turing machine M and an oracle graph G, we are interested in the number of accepting paths of M when given oracle access to G. We define the number of accepting paths of M on input $j \in \mathbb{N}$ (given in binary) with oracle access to G as $\#\mathrm{acc}_{M^G}(j)$. [5] Further we will also call $\#\mathrm{acc}_{M^G}(j)$ a computation. In the following, write $\mathcal{OG}_{=j}$ to denote those graphs in $\mathcal{OG}$ that contain exactly j vertices, $0, \ldots, j-1$ with the natural order on them, for $j \in \mathbb{N}$.

[5] This notation highlights that G is given as an oracle while the number of vertices j is a classical input.

5.20 Definition (Set-instantiator against (M, G, φ))**.** Let M be a nondeterministic Turing machine, let $G \in \mathcal{OG}$ and let φ be an ordered pure graph motif parameter. Let $\top$ be a symbolic top element above the induced subgraph poset $\mathcal{P}(G)$, i.e., $\top \not\sqsubseteq H$ for all induced subgraphs $H \sqsubseteq G$. A *set-instantiator* $\mathcal{SI}$ is a triple of

- some $j \in \mathbb{N}$,
- an instantiation function $\mathsf{inst}_{\mathcal{SI}} : \mathcal{P}(G) \to \mathcal{OG}_{=j}$, and
- a perception function $\mathsf{perc}_{\mathcal{SI}} : \{0,1\}^* \to \mathcal{P}(G) \cup \{\top\}$,

such that both of the following properties hold for all induced subgraphs $H \sqsubseteq G$:

- $\tau \in \{0,1\}^*$ is an accepting path in computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq H$,
- $\varphi(\mathsf{inst}_{\mathcal{SI}}(H))) = \varphi(H)$.

In the above definition, we think of the perception of an accepting computation path of M to be the induced subgraph of G that is observed by M through oracle queries, while computation paths that do not accept are given perception $\top$.

We can now prove by a randomized construction that set-instantiators always exist for polynomial-time NTMs:

5.21 Lemma. *Let $\varphi : \mathcal{OG} \to \mathbb{N}$ be a pure ordered graph motif parameter, let M be a polynomial-time NTM computing* Eval_φ *and let $G \in \mathcal{OG}$. Then there is a set-instantiator against (M, G, φ).*

Proof. For now let $n \in \mathbb{N}$ be undetermined, we will choose it accordingly later. Uniformly at random choose a function $\hat{\xi} : V(G) \to \{0, \dots, n-1\}$ and let it induce a monotone function $\xi : V(G) \to \{0, \dots, |V(G)| \cdot n - 1\}$ by mapping the i-th smallest vertex $v \in V(G)$ to $(i-1) \cdot n + \hat{\xi}(v)$. We then set $j = |V(G)| \cdot n$.

For an ordered graph $H \sqsubseteq G$, we denote by $\xi(H)$ the graph on the vertices $\{0, \dots, |V(G)| \cdot n - 1\}$, where the edges are given as the images of the edges of H under ξ. Any vertex that is not in the image of $\xi(V(H))$ is thus an isolated vertex. We are trying to find some ξ, s.t. for all $H \sqsubseteq G$, all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(H)}}(j)$ do not access the oracle for any edges incident to vertices in $\xi(V(G)) \setminus \xi(V(H))$. By the union bound, if we can show this happening with high probability for each of the finitely many individual $H \sqsubseteq G$, then there is such a ξ, so for now fix some $H \sqsubseteq G$.

Now group the functions ξ by their image of $V(H)$ and fix one such group I. Note that the image of the remaining vertices $V(G) \setminus V(H)$ under ξ is still independently and uniformly distributed from n possible choices each by the choice of $\hat{\xi}$. Consider the set S_I of all vertices queried by all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(H)}}(j)$. Let $t_M(j)$ be the worst-case running-time of M on input j. Each oracle query can query at most two vertices, there are at most $\varphi(H)$ many accepting paths and each accepting path can query the oracle at most $t_M(j)$ many times and as such the size $|S_I|$ is bounded by $2 \cdot \varphi(H) \cdot t_M(j)$, which is polynomial in $\log(j)$ due to M being polynomial-time and H being fixed. As a rough estimate using the Bernoulli inequality, this means that for a fraction of at least $\left(1 - \frac{|S_I|}{n}\right)^{|V(G)|} \geq 1 - \frac{|S_I| \cdot |V(G)|}{n}$ choices of $\hat{\xi}$ resulting in $\xi \in I$, none of the accepting paths queries any edges incident to vertices in $\xi(V(G)) \setminus \xi(V(H))$. By averaging across all possible $\hat{\xi}$ (no longer restricted to I), we see that a fraction of $1 - \frac{\mathrm{polylog}(j) \cdot |V(G)|}{n}$ choices have our desired property.

Using the union bound over all $H \sqsubseteq G$, we get that a fraction of $1 - \frac{\mathrm{polylog}(j) \cdot |V(G)| \cdot 2^{|V(G)|}}{n}$ choices of $\hat{\xi}$ lead to the desired property for all $H \sqsubseteq G$ simultaneously. By choosing n large enough (remember that j and n are polynomially related, so this is always possible), this guarantees the existence of our desired ξ, s.t. for all $H \sqsubseteq G$, all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(H)}}(j)$ do not access the oracle for any edges incident to vertices in $\xi(V(G)) \setminus \xi(V(H))$.

We can now construct our set-instantiator with j chosen as above. The instantiation function for $H \sqsubseteq G$ is $\mathsf{inst}_{\mathcal{SI}}(H) = \xi(H)$. For any computation path $\tau \in \{0,1\}^\star$ of a computation $\#\mathrm{acc}_{M^{\xi(H)}}(j)$, denote by $S_\tau \subseteq \{0, \dots, j-1\}$ the set of vertices that are queried by the computation. We set

$$\mathsf{perc}_{\mathcal{SI}}(\tau) := \begin{cases} G[\xi^{-1}(S_\tau)] & \text{if } \tau \text{ is an accepting path of the computation } \#\mathrm{acc}_{M^{\xi(G)}}(j), \\ \top & \text{otherwise} \end{cases}$$

Note that the preimage $\xi^{-1}(S_\tau)$ is defined to be simply the set $\{v \in V(G) \mid \xi(v) \in S_\tau\}$, even if S_τ is not entirely contained in the image of ξ.

We check the properties of a set-instantiator. Clearly $\varphi(\mathsf{inst}_{\mathcal{SI}}(H)) = \varphi(H)$ for all $H \sqsubseteq G$, since the two graphs are identical except isolated vertices and no pattern in φ contains any isolated vertices. Further we need that for all $H \sqsubseteq G$ we have that $\tau \in \{0,1\}^\star$ is an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq H$. For this fix $H \sqsubseteq G$ and let τ be an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H)}}(j)$. By our choice of ξ we know that no other vertices of $\xi(V(G))\setminus\xi(V(H))$ are queried, or in other words, $\xi^{-1}(S_\tau) \subseteq V(H)$ and thus τ is also an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(G)}}(j)$. This directly implies $\mathsf{perc}_{\mathcal{SI}}(\tau) = G[\xi^{-1}(S_\tau)] \sqsubseteq H$. On the other hand, let $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq H$, then $\mathsf{perc}_{\mathcal{SI}}(\tau) \neq \top$ and τ is an accepting path of the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(G)}}(j)$, since $\mathsf{perc}_{\mathcal{SI}}(\tau) = G[\xi^{-1}(S_\tau)] \sqsubseteq H$, τ is also an accepting path of the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H)}}(j)$. □

If H_1 and H_2 are isomorphic, then it can still be the case that $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H_1)}}(j) \neq \#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H_2)}}(j)$, which we do not want to happen in a good function. For example if G is any graph containing some triangles, then we want the instantiation of every induced triangle of G to have the same number of accepting paths. We thus want to construct an instantiation function that no longer has this problem. We achieve this by creating a very large set-instantiator and then restricting ourselves to some small part of the set-instantiator where we can enforce this property.

Ensuring this structure is achieved by usage of a Ramsey theorem. For two graphs H and G we denote by $\binom{G}{H}$ the subset of induced subgraphs of G that are isomorphic to H.

5.22 Lemma (Main Theorem in [NR77])**.** *Let $F, G \in \mathcal{OG}$ and let $t \in \mathbb{N}$ be fixed. Then there is a $H \in \mathcal{OG}$, such that for every $\Phi : \binom{H}{F} \to \{0,\dots,t\}$, there is a $G_\Phi \sqsubseteq H$ with $G_\Phi \cong G$ and $|\Phi(\binom{G_\Phi}{F})| = 1$.*

Note that Lemma 5.22 is the reason why we need to generalize our result to ordered graphs first instead of being able to prove it directly for unordered graphs. It is known that this type of Ramsey theorem does not hold for unordered graphs [Neš95, Theorem 5.1].

By repeatedly applying Lemma 5.22 we get

5.23 Proposition (Ramsey Theorem)**.** *Let $G \in \mathcal{OG}$ and let $t \in \mathbb{N}$ be fixed. Then there is an $H \in \mathcal{OG}$, such that for every $\Phi : \mathcal{P}(H) \to \{0,\dots,t\}$ there is a $G_\Phi \sqsubseteq H$ with $G_\Phi \cong G$ and, for all $A, B \sqsubseteq G_\Phi$ with $A \cong B$, we have $\Phi(A) = \Phi(B)$.*

This theorem can now be used to ensure that for every polynomial-time NTM there must be some small set of inputs where the NTM behaves like a good function.

5.24 Lemma. *Let φ be a pure ordered graph motif parameter, let M be any nondeterministic polynomial-time Turing machine computing* Eval_φ *and let $G \in \mathcal{OG}$. Then*

there is some $j \in \mathbb{N}$, a function $\mathsf{inst} : \mathcal{P}(G) \to \mathcal{OG}_{=j}$ and a $\mathcal{P}(G)$-good function Ψ with $\#acc_{M^{\mathsf{inst}(F)}}(j) = \Psi(F)$ and $\varphi(\mathsf{inst}(F)) = \varphi(F)$ for every $F \sqsubseteq G$.

Proof. We invoke Proposition 5.23 for G and $t := \max\{\varphi(H) : H \sqsubseteq G\}$ to obtain $\tilde{G} \in \mathcal{OG}$. We now want to color $\mathcal{P}(\tilde{G})$ according to the behaviour of M on the elements of it, in particular how many accepting paths query each specific induced subgraph of $\tilde{G}$. Before we can do this we have to first embed all potential elements of $\mathcal{P}(\tilde{G})$ as oracles. In order to do this we use Lemma 5.21 to construct a set-instantiator $\mathcal{SI}$ against $(M, \tilde{G}, \varphi)$ to obtain a $j \in \mathbb{N}$, $\mathsf{inst}_{\mathcal{SI}}$ and $\mathsf{perc}_{\mathcal{SI}}$ as in Definition 5.20.

For $H \in \mathcal{P}(\tilde{G})$ define $\Phi(H) := \#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(H)}}(j)$. Note that

$$\Phi(H) \;=\; \big|\big\{\tau \in \{0,1\}^* \,:\, \mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq H\big\}\big| = \sum_{F \sqsubseteq H} \big|\big\{\tau \in \{0,1\}^* \,:\, \mathsf{perc}_{\mathcal{SI}}(\tau) = F\big\}\big|. \tag{5.25}$$

Using Φ as a coloring for Proposition 5.23, we obtain G_Φ. Note that $\Phi(H)$ now only depends on the isomorphism type $[H]$ of H for all $H \sqsubseteq G_\Phi$. By induction we see that the number

$$\big|\big\{\tau \in \{0,1\}^* \,:\, \mathsf{perc}_{\mathcal{SI}}(\tau) = H\big\}\big|$$

also depends only on the isomorphism type of H, whenever $H \sqsubseteq G_\Phi$. Denote this number by $\alpha_{[H]}$ and set $\Psi([H]) := \Phi(H)$. This is only well-defined for $H \sqsubseteq G_\Phi$, thus we consider the domain of Ψ to be $\mathcal{P}([G_\Phi]) = \mathcal{P}([G])$. This means that (5.25) simplifies:

$$\Psi([H]) \;=\; \sum_{[F]} \#\mathrm{Ind}([F] \to [H])\alpha_{[F]} \tag{5.26}$$

where the sum is over all isomorphism classes $[F]$ of induced subgraphs $F \sqsubseteq H$. This implies that Ψ is $\mathcal{P}(G)$-good.

It remains to define the function $\mathsf{inst} : \mathcal{P}(G) \to \mathcal{OG}_{=j}$. Let $H \sqsubseteq G$, then $\mathsf{inst}(H) := \mathsf{inst}_{\mathcal{SI}}(H_\Phi)$ for some $H_\Phi \sqsubseteq G_\Phi$ with $H \cong H_\Phi$. The property $\varphi(\mathsf{inst}(H)) = \varphi(H)$ now directly follows from $\mathcal{SI}$ being a set-instantiator. □

Note that Lemma 5.24 only guarantees the local behaviour of M to be $\mathcal{P}(G)$-good. This leaves two possibilities for contradictions: Either the local behaviour is even $(\mathcal{P}(G) \cap \mathcal{OG}^{\mathrm{pure}})$-good, or the function grows too quickly for inputs with many isolated vertices. This next Witness Theorem shows that in the first case we can always find an ordered graph that proves that M does not compute the correct function using a simple linear algebra argument.

5.27 Theorem (The Witness Theorem)**.** *Fix a bad $\varphi : \mathcal{OG}^{pure} \to \mathbb{N}$. Then there exists $G \in \mathcal{OG}^{pure}$ such that for every $(\mathcal{P}(G) \cap \mathcal{OG}^{pure})$-good function $\Psi : (\mathcal{P}(G) \cap \mathcal{OG}^{pure}) \to \mathbb{N}$, there exists $W \in \mathcal{P}(G) \cap \mathcal{OG}^{pure}$ with $\Psi(W) \neq \varphi(W)$.*

Proof. Choose G to be the disjoint union of the graphs in $\operatorname{supp}(\varphi)$ and let $\mathcal{OG}' := \mathcal{P}(G) \cap \mathcal{OG}^{\text{pure}}$, keeping only one representative per isomorphism class. By Fact 5.9, the functions $\#\text{Ind}(H \to \star)$ for $H \in \mathcal{OG}'$ are linearly independent. Thus the patterns and coefficients of any function Ψ with support $\operatorname{supp}(\Psi) \subseteq \mathcal{P}(G) \cap \mathcal{OG}^{\text{pure}}$ are uniquely determined by the evaluations Ψ on $\mathcal{P}(G) \cap \mathcal{OG}^{\text{pure}}$, and there exists a witness $W \in \mathcal{P}(G) \cap \mathcal{OG}^{\text{pure}}$ with $\Psi(W) \neq \varphi(W)$, since Ψ is good while φ is bad (thus, in particular, Ψ and φ are not the same function). □

With this last tool we are finally able to lead the existence of polynomial-time NTMs computing bad ordered graph motif parameters to a contradiction.

5.28 Lemma. *Let M be any nondeterministic polynomial-time Turing machine computing Eval_φ for some bad pure ordered graph motif parameter φ. Then there is a $j \in \mathbb{N}$ and $W \in \mathcal{OG}_{=j}$ such that $\#acc_{M^W}(j) \neq \varphi(W)$.*

Proof. We invoke the Witness Theorem 5.27 with our φ to obtain an $G_1 \in \mathcal{OG}^{\text{pure}}$ as in the theorem. Construct G_2 such that

$$\#\text{Ind}(H \to G_2) > \max\left\{\varphi(F) \,:\, F \in \mathcal{P}(G_1) \cap \mathcal{OG}^{\text{pure}}\right\} \quad \text{for every } H \in \mathcal{P}(G_1) \setminus \mathcal{OG}^{\text{pure}} \tag{5.29}$$

$$\#\text{Ind}(H \to G_2) = \#\text{Ind}(H \to G_1) \quad \text{for every } H \in \mathcal{OG}^{\text{pure}} \tag{5.30}$$

by adding sufficiently many isolated vertices to G_1. For this it is sufficient to add $1 + \max\left\{\varphi(F) \,:\, F \in \mathcal{P}(G_1) \cap \mathcal{OG}^{\text{pure}}\right\}$ isolated vertices between every pair of adjacent vertices in G_1, relative to $\leq_{G_1}$.

We now use Lemma 5.24 to obtain $j \in \mathbb{N}$, a function $\mathsf{inst} : \mathcal{P}(G_2) \to \mathcal{OG}_{=j}$ and a $\mathcal{P}(G_2)$-good function Ψ with $\#\text{acc}_{M^{\mathsf{inst}(H)}}(j) = \Psi(H)$ and $\varphi(\mathsf{inst}(H)) = \varphi(H)$ for every $H \sqsubseteq G_2$. There are now two possibilities to find the required counterexample W: If there are no basis functions present in Ψ with a pattern in $\mathcal{P}(G_1) \setminus \mathcal{OG}^{\text{pure}}$, we can invoke the Witness Theorem again, however if any of these basis functions are present we can construct a direct counterexample.

We first handle the case where such a basis function is present, so let the coefficient of $\#\text{Ind}(H \to \star)$ in Ψ be positive for some $H \in \mathcal{P}(G_1) \setminus \mathcal{OG}^{\text{pure}}$. Then

$$\Psi(G_2) \geq \#\text{Ind}(H \to G_2) > \max\left\{\varphi(F) \,:\, F \in \mathcal{P}(G_1) \cap \mathcal{OG}^{\text{pure}}\right\} \geq \varphi(G_1) = \varphi(G_2)$$

by (5.29) and (5.30), combined with the fact that φ is pure, and our counterexample is $W := G_2$.

Otherwise assume that no such basis function is present. Then $\tilde{\Psi}$ obtained from Ψ by restricting to $\mathcal{P}(G_1)$ (i.e. setting all coefficients of $\mathcal{P}(G_2) \setminus \mathcal{P}(G_1)$ to zero), is $(\mathcal{P}(G_1) \cap \mathcal{OG}^{\text{pure}})$-good, which is a requirement for the second part of the Witness Theorem 5.27 (we already used it to obtain G_1). Furthermore $\tilde{\Psi}_{|\mathcal{P}(G_1)} = \Psi_{|\mathcal{P}(G_1)}$. We invoke the

second part of the Witness Theorem 5.27 to find a point $W \in \mathcal{P}(G_1) \cap \mathcal{OG}^{\text{pure}}$ with $\Psi(W) = \tilde{\Psi}(W) \neq \varphi(W)$ which is our counterexample.

Independent of how we have obtained our W we observe $\#\text{acc}_{M^{\mathsf{inst}(W)}}(j) = \Psi(W) \neq \varphi(W) = \varphi(\mathsf{inst}(W))$, which completes the proof for $W' = \mathsf{inst}(W)$. □

This proves the separation in Theorem 5.17, which together with Lemmas 5.12 and 5.13 finishes the proof of the theorem.

5.6 Overview of Results for Relational Structures and Categories

Induced subgraph counts can also be considered in a colored setting, where each vertex has a color and colored graphs are considered isomorphic if the colors of the vertices that are mapped to each other by the isomorphism agree. Such colored patterns occur, e.g., within the proofs of some parameterized hardness results [DRSW22; Cur24], as access to vertex-colors gives more versatility in the reduction. It is natural to ask whether our main theorem generalizes from graphs to colored graphs and to other more general versions of graphs, such as hypergraphs of bounded edge-cardinality. We answer this question positively for general *relational structures* in Section 5.7. The proof proceeds along the same lines as Theorem 5.14, our main theorem about graphs.

Generalizing further from relational structures, we consider the vastly more general setting of *category theory* in Section 5.8 and show a variant of Theorem 5.14 for motif parameters defined over categories. It turns out that the arguments from the main proof can be adapted for this purpose, but only after significant technical effort: While homomorphisms and embeddings naturally generalize from graphs to categories, nontrivial concepts from category theory are needed to formulate the proper requirements on categories that allow us to handle subobjects similarly to induced subgraphs. Among other such requirements, we require *well-powered* categories (where every subobject itself has only finitely many isomorphism types of subobjects) and *proper factorization systems* (that decompose morphisms into compositions of epimorphisms and monomorphisms). We then use the general category-theoretic framework to study local combinatorial interpretations of counting problems related to finite vector spaces and to so-called *parameter sets*, which play a role in enumerative combinatorics.

5.7 Relational Structures

First, we need to transfer some terminology from graphs to relational structures and introduce some additional notions that are specific for relational structures.
Let $k_1, \ldots, k_\ell > 0$. A finite relational structure of *type* $\Delta := (k_1, \ldots, k_\ell)$ consists of a

finite universe V of individuals (which we call vertices, in analogy to our terminology for graphs) and relations $R_1, \dots, R_\ell$ with $R_i \subseteq \binom{V}{k_i}$. Denote the class of all finite relational structures of type Δ as $\mathrm{Rel}(\Delta)$. From now on, we will refer to finite relational structures simply as relational structures. For any relational structure $A = (V, R_1, \dots, R_\ell)$, we write $V(A) := V$ and $R_i(A) = R_i$ for $i = 1 \dots \ell$.

A relational structure A is an *induced substructure* of B, written as $A \sqsubseteq B$, if $V(A) \subseteq V(B)$ and $R_i(A) = R_i(B) \cap \binom{V(A)}{k_i}$ for all $1 \leq i \leq \ell$. Furthermore A is a *substructure* of B, written as $A \subseteq \mathrm{B}$, if $V(A) \subseteq V(B)$ and $R_i(A) \subseteq R_i(B) \cap \binom{V(A)}{k_i}$ for all $1 \leq i \leq \ell$. A function $f : V(A) \to V(B)$ is called a *strong embedding* if f is injective and $X \in R_i(A) \Leftrightarrow f(X) \in R_i(B)$ for all $1 \leq i \leq \ell$ and $X \subseteq V(A)$.

A relational structure A is called *irreducible* if for all different $u, v \in V(A)$ there is some $1 \leq i \leq \ell$ and $X \in R_i(A)$ with $u, v \in X$.

A vertex $v \in V(A)$ is called P-padding for $P = (p_1, \dots, p_\ell) \in \{0,1\}^\ell$ if for all $p_i = 0$ we have $\{X \in R_i(A) : v \in X\} = \emptyset$ and for all $p_i = 1$ we have $\{X \in R_i(A) : v \in X\} = \{X \in \binom{V(A)}{k_i} : v \in X\}$, i.e., v is not in a relation with any other vertices for $p_i = 0$ and in all possible relations with other vertices for $p_i = 1$. A relational structure A is now called P-pure if it does not contain any P-padding vertices.

A subset $F \subseteq \mathrm{Rel}(\Delta)$ is called *upwards closed* if for every $A \in F$ and all $B \in \mathrm{Rel}(\Delta)$ with $V(A) = V(B)$ and $A \subseteq B$ we have $B \in F$. Let $F \subseteq \mathrm{Rel}(\Delta)$ be a (potentially empty) upwards closed set of P-pure irreducible relational structures of type Δ. Now let $U = \mathrm{Forb}(F)$ be the set of all relational structures of type Δ that do not contain any substructure isomorphic to any element in F. This is equivalent to U not containing any induced substructures isomorphic to elements in F, due to F being upwards closed. Restricting U to only the P-pure relational structures yields $U^{P\text{-pure}}$.

As was the case for graphs, the Ramsey theorem only holds for an ordered version of relational structures. We say $\mathrm{ORel}(\Delta)$ is the class of all ordered relational structures of type Δ. Ordered relational structures are defined, as in the graph case, by incorporating a linear order $\leq_V$ on the universe into the structure and extending notions of homomorphisms, (strong) embeddings, and isomorphisms to respect the ordering, which also yields notions of (induced) substructures for ordered structures. For a relational structure A and $X \subseteq V(A)$, write $A[X]$ for the substructure induced by X. Parallel to Definition 5.10, define (ordered) motif parameters φ to be linear combinations of induced substructure counts $\#\mathrm{Ind}(A \to \star)$ for fixed (ordered) relational structures A. We call φ P-pure if all its patterns are P-pure. For a class of relational structures U, we speak of a U motif parameter if its domain is restricted to U.

Relational structures A with $V(A) = \{0, \dots, j-1\}$ are encoded as oracles (x, O) by choosing $x = j$ (encoded in binary), and by querying O with $(i, (v_1, \dots, v_{k_i}))$, a Turing machine can ask whether $\{v_1, \dots, v_{k_i}\} \in R_i(A)$.

We can now state our main theorem, in analogy to Theorem 5.17.

5.31 Theorem. *Let Δ be a type, let $P \in \{0,1\}^\ell$ and let $F \in \mathrm{ORel}(\Delta)$ be an upwards closed subset of P-pure irreducible ordered relational structures.*

Let $\varphi : \mathrm{Forb}(F) \to \mathbb{N}$ be a P-pure ordered $\mathrm{Forb}(F)$ motif parameter. Then $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ iff all coefficients of φ in the $\#\mathrm{Ind}(A_i \to \star)$ basis are nonnegative integers.

We remind that Eval_φ is a promise problem. As such the NTM only has to count correctly whenever the pair (x, O) correctly encodes an ordered relational structure from the domain of φ, in this case $U = \mathrm{Forb}(F)$. This for example implies that for $\Delta = (2)$, $P = (0)$ and $F = \{K_3\}$ (the clique on 3 vertices), the counter example oracle input for each NTM will itself be a graph without any triangles.

In the following, fix Δ, P, F and $U = \mathrm{Forb}(F)$ as in Theorem 5.31.

The proof of Theorem 5.31 is almost identical to that of Theorem 5.17, so we will only point out some specific modifications required in the proof:

- The Ramsey theorem: Lemma 5.22 was shown in [NR77] not only for the class of all finite graphs, but more generally for the class of all relational structures defined via a set of irreducible forbidden substructures, such as U.
- The value of a P-pure ordered motif parameter φ does not change by adding or removing P-padding vertices, since any P-padding vertex v in an ordered relational structure stays a P-padding vertex in any induced substructure containing v. Anywhere we were filling up a graph with isolated vertices, we instead add P-padding vertices. (Note that isolated vertices are simply (0)-padding vertices.)
- Since all elements of F are P-pure, it is impossible to leave U by adding P-padding vertices to a relational structure that was in U.
- Due to the changes of the way we store an ordered relational structure as an oracle, we will be showing the existence of set-instantiators explicitly again.

The notion of a set-instantiator is generalized as follows. We write $U_{=j}$ for the ordered relational structures in U whose universes are exactly the vertices $\{0, \ldots, j-1\}$ with the natural order on them.

5.32 Definition (Set-instantiator against (M, A, φ))**.** Let M be a nondeterministic Turing machine, let $A \in U_{=j}$ and let φ be an ordered P-pure motif parameter.

Let $\top$ be a symbolic top element above the induced substructure poset $\mathcal{P}(A)$, i.e., $\top \not\sqsubseteq B$ for all induced substructures $B \sqsubseteq A$. A *set-instantiator* $\mathcal{SI}$ is a triple of

- some $j \in \mathbb{N}$,
- an instantiation function $\mathsf{inst}_{\mathcal{SI}} : \mathcal{P}(A) \to U_{=j}$, and
- a perception function $\mathsf{perc}_{\mathcal{SI}} : \{0,1\}^* \to \mathcal{P}(A) \cup \{\top\}$,

such that the following property holds for all induced substructures $B \sqsubseteq A$:

- $\tau \in \{0,1\}^*$ is an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(B)}}(j)$ if and only if $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq B$, and
- $\varphi(\mathsf{inst}_{\mathcal{SI}}(B))) = \varphi(B)$.

5.33 Lemma. *Let $\varphi : U \to \mathbb{N}$ be a P-pure ordered U motif parameter, let M be a polynomial-time NTM computing Eval_φ and let $A \in U$. Then there is a set-instantiator against (M, A, φ).*

Proof. For now let $n \in \mathbb{N}$ be undetermined, we will choose it accordingly later. Uniformly at random choose a function $\hat{\xi} : V(A) \to \{0, \dots, n-1\}$ and let it induce a monotone function $\xi : V(A) \to \{0, \dots, |V(A)| \cdot n - 1\}$ by mapping the i-th smallest vertex $v \in V(A)$ to $(i-1) \cdot n + \hat{\xi}(v)$. We then set $j = |V(A)| \cdot n$.

For an ordered relational structure $B \sqsubseteq A$, we denote by $\xi(B)$ the relational structure on the vertices $\{0, \dots, |V(G)| \cdot n - 1\}$, where the relations are given as the images of the relations of B under ξ. Furthermore, for every vertex v that is not in the image of $\xi(V(A))$, make v a P-padding vertex. We are trying to find some ξ, s.t. for all $B \sqsubseteq A$, all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(B)}}(j)$ do not access the oracle for any relations incident to vertices in $\xi(V(A)) \setminus \xi(V(B))$. By the union bound, if we can show this happening with high probability for each of the finitely many individual $B \sqsubseteq A$, then there is such a ξ, so for now fix some $B \sqsubseteq A$.

Now group the functions ξ by their image of $V(B)$ and fix one such group I. Note that the image of the remaining vertices $V(A) \setminus V(B)$ under ξ is still independently and uniformly distributed from n possible choices each by the choice of $\hat{\xi}$. Consider the set S_I of all vertices queried by all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(B)}}(j)$. Let $t_M(j)$ is the worst-case running-time of M on input j. Each oracle query queries one of the relations at a time and thus at most $\max(k_1, \dots, k_\ell)$ vertices, there are at most $\varphi(B)$ many accepting paths and each accepting path can query the oracle at most $t_M(j)$ many times and as such the size $|S_I|$ is bounded by $\max(k_1, \dots, k_\ell) \cdot \varphi(B) \cdot t_M(j)$, which is polynomial in $\log(j)$ due to M being polynomial-time and B and $k_1, \dots, k_\ell$ being fixed. As a rough estimate using the Bernoulli inequality and $|V(A) \setminus V(B)| \leq |V(A)|$, this means that for a fraction of at least $\left(1 - \frac{|S_I|}{n}\right)^{|V(A)|} \geq 1 - \frac{|S_I| \cdot |V(A)|}{n}$ choices of $\hat{\xi}$ resulting in $\xi \in I$, none of the accepting paths queries any edges incident to vertices in $\xi(V(A)) \setminus \xi(V(B))$. By averaging across all possible $\hat{\xi}$ (no longer restricted to I), we see that a fraction of $1 - \frac{\mathrm{polylog}(j) \cdot |V(A)|}{n}$ choices have our desired property.

Using the union bound over all $B \sqsubseteq A$, we get that a fraction of $1 - |\mathcal{P}(A)| \cdot \frac{\mathrm{polylog}(j) \cdot |V(A)|}{n}$ choices of $\hat{\xi}$ lead to the desired property for all $B \sqsubseteq A$ simultaneously. By choosing n large enough (remember that j and n are polynomially related, so this is always possible), this guarantees the existence of our desired ξ, s.t. for all $B \sqsubseteq A$, all accepting paths of the computation $\#\mathrm{acc}_{M^{\xi(B)}}(j)$ do not access the oracle for any edges incident to vertices in $\xi(V(A)) \setminus \xi(V(B))$.

We can now construct our set-instantiator with j chosen as above. The instantiation function for $B \sqsubseteq A$ is

$$\mathsf{inst}_{\mathcal{SI}}(B) := \xi(B)\,.$$

For any computation path $\tau \in \{0,1\}^\star$ of a computation $\#\mathrm{acc}_{M^{\xi(B)}}(j)$, denote by $S_\tau \subseteq \{0,\ldots,j-1\}$ the set of vertices that are queried by the computation. We set

$$\mathsf{perc}_{\mathcal{SI}}(\tau) := \begin{cases} A[\xi^{-1}(S_\tau)] & \text{if } \tau \text{ is an accepting path of the computation } \#\mathrm{acc}_{M^{\xi(A)}}(j), \\ \top & \text{otherwise} \end{cases}$$

It remains to check the properties of a set-instantiator. Clearly $\varphi(\mathsf{inst}_{\mathcal{SI}}(B)) = \varphi(B)$ for all $B \sqsubseteq G$, since the two relational structures are identical except P-padding vertices and no pattern in φ contains any P-padding vertices. Further we need that for all $B \sqsubseteq A$ we have that $\tau \in \{0,1\}^\star$ is an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(B)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq B$. For this fix $B \sqsubseteq A$ and let τ be an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(B)}}(j)$. By our choice of ξ we know that no other vertices of $\xi(V(A))\setminus\xi(V(B))$ are queried, or in other words, $\xi^{-1}(S_\tau) \subseteq V(B)$ and thus τ is also an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(A)}}(j)$. This directly implies $\mathsf{perc}_{\mathcal{SI}}(\tau) = A[\xi^{-1}(S_\tau)] \sqsubseteq B$. On the other hand, let $\mathsf{perc}_{\mathcal{SI}}(\tau) \sqsubseteq B$, then $\mathsf{perc}_{\mathcal{SI}}(\tau) \neq \top$ and τ is an accepting path of the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(A)}}(j)$. Since $\mathsf{perc}_{\mathcal{SI}}(\tau) = A[\xi^{-1}(S_\tau)] \sqsubseteq B$, τ is also an accepting path of the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(B)}}(j)$. □

5.7.1 Directed Relational Structures

So far, for all of our graphs and relational structures we have assumed that they are undirected and have no self-loops or relations with duplicate vertices.

We study two further generalizations to ordered and unordered relational structures A, where each relation R_i can now be one of following different variants:

1. Set relations, i.e. $R_i \subseteq \binom{V}{k_i}$ which is the kind of relational structures we have studied so far.

2. Multiset relations, i.e. R_i is a set of multiset of vertices from V with k_i total (not necessarily distinct) elements each. These for example would allow us to express undirected self-loops in the graph setting.

3. List relations without repetitions, i.e. R_i is a tuple of k_i vertices from V, but without repetitions. Directed graphs without self-loops fall under this category.

4. List relations with repetitions, i.e. R_i is a tuple of k_i elements from V with repetitions. Directed graphs with self-loops fall under this category.

We can even allow different relations to be of different variants. We call such an (ordered) relational structure a mixed (ordered) relational structure. A *mixed type* $\tilde{\Delta}$ is now a type Δ, combined with the information about which variant each relation uses. The class $\mathrm{MORel}(\tilde{\Delta})$ is then the class of all mixed ordered relational structures of type $\tilde{\Delta}$. The induced substructures of a mixed (ordered) relational structure are now defined in the natural way and homomorphisms and (strong) embeddings preserve the order and multiplicities of the mixed relations. We call a mixed (ordered) relational structure A irreducible if for every different $u, v \in V(A)$ there is some $1 \leq i \leq \ell$ and $X \in R_i(A)$, such that X contains both u and v (as a set, multiset, or list, depending on the variant of R_i). For a vertex v to be a P-padding vertex the relation R_i needs to contain all possible entries including v if $p_i = 1$ and for $p_i = 0$ it has to contain none of the entries including v. As usual A is called P-pure if it contains no P-padding vertices.

In an ordered relational structure, each of these types of relations can be expressed by using one or more set relations in the following way.

Multiset relation: Replace R_i by one set relation $(R_i)_C$ of arity $|C|$ for every composition C of k_i. Let e be a hyperedge in R_i with vertices $v_1, \ldots, v_\nu$ with multiplicities $\lambda_1, \ldots, \lambda_\nu$. W.l.o.g. assume that $v_1 \leq v_2 \leq \ldots \leq v_\nu$ according to the linear order on V. Then $\{v_1, \ldots, v_\nu\}$ is added to $(R_i)_{(\lambda_1,\ldots,\lambda_\nu)}$.

List relation without repetition: Replace R_i by one set relation $(R_i)_\prec$ of arity k_i for every possible total order $\prec$ of $\{1, \ldots, k_i\}$. Let $(v_1, \ldots, v_{k_i}) \in R_i$. Define the order $\prec_{(v_1,\ldots,v_{k_i})}$ on $\{1, \ldots, k_i\}$ via $i \prec j := v_i \leq v_j$ and add $\{v_1, \ldots, v_{k_i}\}$ to $(R_i)_{\prec_{(v_1,\ldots,v_{k_i})}}$.

List relation with repetition: Same as above, however $\prec$ is now a total preorder, i.e. it does not have to be anti-symmetric. This also implies that not all of the added relations have arity exactly k_i, but rather arity $\leq k_i$, similarly to the multiset relation case.

Denote the resulting type of the ordered relational structures as Δ'. Note that Δ' only depends on $\tilde{\Delta}$. We call the function applying the above conversion rules $\mathsf{conv} : \mathrm{MORel}(\tilde{\Delta}) \to \mathrm{ORel}(\Delta')$.

5.34 Fact. This construction is bijective, i.e. the conversion mapping conv is bijective. Furthermore, both conv and conv^{-1} are efficiently computable in the sense that it is possible, in polynomial time in $\log(|V(A)|)$, to compute the image (or preimage) of any possible entry of the relations under conv.

5.35 Fact. Moreover conv commutes with (strong) embeddings and thus induced subgraphs and isomorphisms. In particular this directly implies that $\#\mathrm{Ind}(B \to A) = \#\mathrm{Ind}(\mathsf{conv}(B) \to \mathsf{conv}(A))$.

These two facts also imply a further generalization of the Ramsey theorem in [NR77] to mixed ordered relational structures by applying conv before using the Ramsey theorem on ordered relational structures and then applying conv^{-1} to convert the resulting relational

structure back to a mixed ordered relational structure. However we do not use this variant of the Ramsey theorem here and instead do a direct reduction.

5.36 Theorem. *Let $\tilde{\Delta}$ be a mixed type with ℓ relations, let $P \in \{0,1\}^\ell$ and let $F \subseteq \mathrm{MORel}(\tilde{\Delta})$ be an upwards closed subset of P-pure irreducible mixed ordered relational structures.*

Let $\varphi : \mathrm{Forb}(F) \to \mathbb{N}$ be a P-pure ordered $\mathrm{Forb}(F)$ motif parameter, i.e., $\varphi(B) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(A_i \to B)$ for pairwise non-isomorphic P-pure $A_1, \ldots, A_s$ and coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$. Then the evaluation problem Eval_φ is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ iff all coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$ are nonnegative integers.

Proof. Let $F' = \{\mathsf{conv}(A) : A \in F\}$ and let P' be obtained from P by setting $p'_{j_i} = p_i$ where j_i ranges over all the relations added by the conversion from relation i. Then F' is upwards closed and only contains P'-pure ordered relational structures. We now define $\varphi' : \mathrm{Forb}(F') \to \mathbb{N}$ as $\varphi'(B') = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(\mathsf{conv}(A_i) \to B')$. All the $\mathsf{conv}(A_i)$ are pairwise non-isomorphic and P'-pure. By Theorem 5.31 we know that $\mathrm{Eval}_{\varphi'} \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ iff $\alpha_1, \ldots, \alpha_s$ are nonnegative integers. Given an oracle encoding some $B \in \mathrm{Forb}(F)$ we can now in polynomial time simulate an oracle encoding $\mathsf{conv}(B) \in \mathrm{Forb}(F')$ by using Fact 5.34. Moreover by Fact 5.35 $\varphi(B) = \varphi'(\mathsf{conv}(B))$, thus if $\mathrm{Eval}_{\varphi'} \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$, then also $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$.

Since conv is bijective and efficiently computable in both directions this argument also reverses and proves $\mathrm{Eval}_{\varphi'} \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot} \Leftrightarrow \mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$. □

In the same way that we have proven Theorem 5.14 from Theorem 5.17, we also get a variant with unordered mixed relational structures.

5.37 Theorem. *Let $\tilde{\Delta}$ be a mixed type with ℓ relations, let $P \in \{0,1\}^\ell$ and let $F \subseteq \mathrm{MRel}(\tilde{\Delta})$ be an upwards closed subset of P-pure irreducible mixed relational structures.*

Let $\varphi : \mathrm{Forb}(F) \to \mathbb{N}$ be a P-pure $\mathrm{Forb}(F)$ motif parameter, i.e., $\varphi(B) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(A_i \to B)$ for pairwise non-isomorphic P-pure $A_1, \ldots, A_s$ and some coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$. Then the evaluation problem Eval_φ is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot\cdot\cdot}$ iff all the coefficients $\alpha_1, \ldots, \alpha_s$ are nonnegative integers.

Proof. We observe

$$\begin{aligned}&\#\mathrm{Ind}((V_i, R_{i,1}, \ldots, R_{i,\ell}) \to (V, R_1, \ldots, R_\ell))\\ &= \sum_{\leq_{V_i}} \#\mathrm{Ind}((V_i, R_{i,1}, \ldots, R_{i,\ell}, \leq_{V_i}) \to (V, R_1, \ldots, R_\ell, \leq_V))\end{aligned}$$

where $\leq_V$ is any arbitrary linear ordering of V and $\leq_{V_i}$ sums over all linear orders of V_i that result in non-isomorphic ordered relational structures. Further let $F' \subseteq \mathrm{MORel}(\tilde{\Delta})$ be obtained from the elements of F combined with all possible linear orders of their

vertices. Note that all elements of F' are irreducible and P-pure and F' is still upwards closed, we can thus invoke Theorem 5.36.

The same argument as for Theorem 5.14 finishes the proof. □

5.7.2 Applications

For colored graphs, a special case of relational structures, we get a particularly clean dichotomy, here we do not rely on the patterns not containing any isolated vertices, we only require that there is some color that can be used as a color in the target graph that does not appear in any of the pattern graphs. Let $\mathrm{Col}\mathcal{G}(c+1)$ to be the set of all colored graphs on $c+1$ colors. We rephrase $\mathrm{Col}\mathcal{G}(c+1)$ as relational structures with $\Delta = (2, \underbrace{1, \ldots, 1}_{c \text{ times}})$, by adding a unary relation for every color, except the color 0. By choosing $P = (0, \ldots, 0)$ we set P-padding vertices to be exactly those vertices using the color 0. A colored graph is thus P-pure iff it contains no vertices with color 0. Furthermore we set F to be the set of relational structures on a single vertex, where at least two different of the unary relations are set. These correspond to vertices that are assigned more than one color, turning $\mathrm{Forb}(F)$ into exactly the set of colored graphs where each vertex has exactly one color, either one of $1, \ldots, c$ or the color 0 by absence of any other color.

By using Theorem 5.37 we thus get

5.38 Theorem. *Let $\varphi : \mathrm{Col}\mathcal{G}(c+1) \to \mathbb{N}$ be a colored graph motif parameter, i.e., $\varphi(G) = \sum_{i=1}^{s} \alpha_i \cdot \#\mathrm{Ind}(H_i \to G)$ for pairwise non-isomorphic graphs $H_1, \ldots, H_s$ using the colors $\{1, \ldots, c\}$ and coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$. Then the evaluation problem Eval_φ is in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ iff all coefficients $\alpha_1, \ldots, \alpha_s \in \mathbb{Q}$ are nonnegative integers.*

Note that while we restrict the number of colors within the theorem itself, the same result holds true if the number of colors is allowed to grow with φ, by simply choosing the number of colors for the target graph to be one bigger than the fixed number of colors used in the patterns of φ.

5.8 Categorical Generalization

In the proofs of Section 5.5 we didn't use particularly many properties of ordered graphs to show our lower bound, except some structure of (induced) subobjects, a Ramsey property and the existence of set-instantiators. The goal of this section is now to generalize the proof and make it more explicit which properties in detail are needed for our proof to work.

5.8.1 Basic Category Theory

We follow the definitions of [Rie17] and refer the reader to it should they want more than just these basic definitions. A category C consists of two collections:

- A collection of *objects*.
- A collection of *morphisms* between objects.

such that:

- Every morphism f has a specified *domain* and a *codomain* object. We write $f : a \to b$ if f has domain a and codomain b. Additionally we write $\operatorname{dom} f = a$ and $\operatorname{cod} f = b$.
- For every object a there is an *identity morphism* $\mathrm{id}_a : a \to a$.
- For every two morphisms $f : a \to b$ and $g : b \to c$, there is a specific *composite morphism* $gf : a \to c$.
- For every morphism $f : a \to b$ both $f\mathrm{id}_a$ and $\mathrm{id}_b f$ are identical to f, i.e. the identity morphisms are the neutral elements of composition.
- For every triple of morphisms $f : a \to b$, $g : b \to c$ and $h : c \to d$ we have that $h(gf)$ and $(hg)f$ are identical, i.e. composition is associative.

A morphism $f : b \to c$ is called

- a *monomorphism*, if for any morphisms $g, h : a \to b$ with $fg = fh$ we already have $g = h$. We also say that f is monic or a mono.
- an *epimorphism*, if for any morphisms $g, h : c \to d$ with $gf = hf$ we already have $g = h$. We also say that f is epic or an epi.
- an *isomorphism*, if there is a morphism $g : c \to b$ with $fg = \mathrm{id}_c$ and $gf = \mathrm{id}_b$. In this case we call b and c *isomorphic*, also written as $b \cong c$.

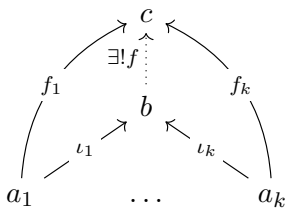
If for every two objects a, b the collection of all morphisms from a to b is a set[6], then C is called *locally small*. If this collection is even finite, then it is called *locally finite*.

We call some category D a *subcategory* of C if the objects of D are a subcollection of the objects of C and the morphisms of D are a subcollection of the morphisms of C. Note that this includes all relevant identity and composite morphisms in order for D to be a category. We say D is a *full subcategory* of C if, additionally, for every two objects a, b in D, it contains all morphisms with domain a and codomain b from C.

The last basic concept we need is the concept of finite coproducts. For objects $a_1, \ldots, a_k$ we say that an object b, together with morphisms $\iota_1, \ldots, \iota_k$ where $\iota_i : a_i \to b$, called the *coproduct injections*[7], is the *coproduct* of $a_1, \ldots, a_k$, if for all objects c and all morphisms $f_1, \ldots, f_k$ with $f_i : a_i \to c$ there is a unique morphism $f : b \to c$ such that the following diagram commutes:

[6] We don't want to to get into the set versus class discussion too much here. While some of our categories have a class worth of objects, they are essentially small, meaning there is only a set worth of non-isomorphic objects. Further all of our categories are locally small.

[7] These coproduct injections do not have to be injective, nor mono



One can show that the object b is unique up to isomorphism, so we write $\bigsqcup_{i=1}^{k} a_i$ for it. We also write $\bigsqcup_{i=1}^{k} f_i$ for the morphism f.

5.8.2 Counting in Categories

Our objects of consideration are now objects of some category C. To specify the subobjects we are counting, we designate a class of morphisms from C:

5.39 Definition. Let C be a locally small category and let $\mathcal{M}$ be some class of morphisms in C that contains all isomorphisms and is closed under composition. For any objects a, b we call morphisms $a \to b$ from $\mathcal{M}$ the $\mathcal{M}$-subobjects of a under b. We identify morphisms $f, f' : a \to b$ if there is an isomorphism $g : a \to a$ with $f' = fg$. The set of all $\mathcal{M}$-subobjects of a under b is denoted by $\mathrm{Sub}_{\mathcal{M}}(a \to b)$, while the class of all $\mathcal{M}$-subobjects under b is denoted by $\mathcal{P}(b)$. If all such $\mathcal{P}(b)$ are small (i.e. a set), then we call C $\mathcal{M}$-well-powered and if they are all finite C is called finitely $\mathcal{M}$-well-powered. For notational convenience, we write $\mathrm{dom}\,\mathcal{P}(b) := \{\mathrm{dom}\, f \,:\, f \in \mathcal{P}(b)\}$.

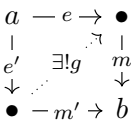
Let $f : a \to b$ and $f' : a' \to b$ be $\mathcal{M}$-subobjects under b. We write $f' \prec f$ if there is some morphism g with $f' = fg$.

The class $\mathcal{M}$ in here should be thought of as some class of special monomorphisms. In the example of undirected graphs, $\mathcal{M}$ is the set of strong graph embeddings, i.e. injective graph homomorphisms that preserve both edges and non-edges. The poset $\mathcal{P}(G)$ is then the set of all induced subgraphs of G (represented via their graph embeddings), while $\mathrm{dom}\,\mathcal{P}(G)$ contains all graphs with isomorphism types that are among the induced subgraphs of G.

Combined with a second class of morphisms $\mathcal{E}$, we obtain a factorization system:

5.40 Definition. Let C be a category and let $\mathcal{E}$ and $\mathcal{M}$ be some classes of morphisms in C. We call $(\mathcal{E}, \mathcal{M})$ a factorization system if they fulfil the following properties:

1. Every morphism $f : a \to b$ factors as $f = me$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$. This factorization is unique, up to a unique isomorphism, i.e. if there is some other factorization $f = m'e'$ with $e' \in \mathcal{E}$ and $m' \in \mathcal{M}$, then there is a unique isomorphism g such that the following diagram commutes:



2. Both $\mathcal{E}$ and $\mathcal{M}$ contain all isomorphisms and are closed under composition.

A factorization system is called proper, if all morphisms in $\mathcal{E}$ are epi and all morphisms in $\mathcal{M}$ are mono.

One example of such a factorization system in the category of finite undirected graphs is given by choosing $\mathcal{E}$ as the (vertex) surjective graph homomorphisms, and $\mathcal{M}$ as the strong graph embeddings. The $\mathcal{M}$-subobjects of H in G are precisely the induced subgraphs of G that are isomorphic to H. See Section 5.8.3 for more details on this.

Except the properties directly visible in Definition 5.40, we need one derived property (see [Rie08][Lemma 1.13] for a proof):

5.41 Fact (Cancellation Properties). Let $(\mathcal{E}, \mathcal{M})$ be a factorization system and $f : a \to b$ and $g : b \to c$ be morphisms. Then we have the following two implications:

- If $gf \in \mathcal{E}$ and $f \in \mathcal{E}$, then $g \in \mathcal{E}$.
- If $gf \in \mathcal{M}$ and $g \in \mathcal{M}$, then $f \in \mathcal{M}$.

Note that this implies that in a $(\mathcal{E}, \mathcal{M})$ factorization system, for $\mathcal{M}$-subobjects f and f', whenever $f' \prec f$, i.e. if there is some morphism g with $f' = fg$, then g is already in $\mathcal{M}$. In other words: Being a $\mathcal{M}$-subobject is well-behaved under composition and cancellation.

Further, in the cases that interest us, there are no non-trivial morphisms $f : a \to a$ in $\mathcal{M}$:

5.42 Lemma. *Let C be a locally small, finitely $\mathcal{M}$-well-powered category with a proper $(\mathcal{E}, \mathcal{M})$-factorization system. Then any morphism $f : a \to a$ in $\mathcal{M}$ is already an isomorphism.*

Proof. We consider the set of morphisms $\{f^n : n \in \mathbb{N}\}$. Since $\mathcal{M}$ is closed under composition, all these morphisms are in $\mathcal{M}$ and thus are $\mathcal{M}$-subobjects of a. By the pigeonhole principle, combined with C being finitely $\mathcal{M}$-well-powered, there must be some $n < m \in \mathbb{N}$ with f^n and f^m being the same $\mathcal{M}$-subobject, i.e. $f^n = f^m g$ with some isomorphism $g : a \to a$. This leads to the following diagram with $f^n \mathrm{id}_a = f^m g$:

$$a \underset{\mathrm{id}_a}{\overset{f^{m-n}g}{\rightrightarrows}} a \xrightarrow{f^n} a$$

Due to $f^n \in \mathcal{M}$ and the factorization system being proper, we conclude that f^n is a monomorphism and $f^{m-n}g = \mathrm{id}_a$, which is equivalent to $f^{m-n} = g^{-1}$. It is now obvious that $f^{m-n-1}g = gf^{m-n-1}$ is the inverse of f. □

Formally, it is not clear how to attach an object from some category C as an oracle to an NTM. In order to do so, we denote by $C_{=j}$ some set of objects from C that can be encoded as a polylog(j)-bit oracle, i.e. as a subset of $\{0,1\}^{\mathrm{bitsize}(j)}$ for some efficiently computable function bitsize $\in$ polylog(j). We require that for every object c in C, there is some $j \in \mathbb{N}$ and a $c' \in C_{=j}$, such that c and c' are isomorphic.

Given a nondeterministic Turing machine M and an oracle object $c \in C_{=j}$, we are interested in the number of accepting paths of M when given oracle access to the j-bit oracle encoding c. We define the number of accepting paths of M on input $j \in \mathbb{N}$ (given in binary[8]) with oracle access to c as $\#\mathrm{acc}_{M^c}(j)$. Further we will also call $\#\mathrm{acc}_{M^c}(j)$ a computation.

[8] The precise nature of how j is given, what the bitsize of the encodings of elements from $C_{=j}$ is and even whether the oracle is binary or not does not matter. The main importance is, that in polynomial time in the length of the input, M can query the oracle at polynomially many positions out of exponentially many possible positions.

Our setup has two categories C and P, the ambient category and the category of *pure* objects respectively. While we list here most of the general properties of both categories we assume in general, each lemma still explicitly lists the properties it uses to make them easier to follow. Both categories are locally small and P is a full subcategory[9] of C, closed under isomorphisms, i.e. if a and b are isomorphic objects in C, they are either both included in P or neither of them is. Moreover there is a compatible proper $(\mathcal{E}, \mathcal{M})$-factorization system of C and P. By compatible we mean some slight abuse of notation: Not all morphisms in $\mathcal{M}$ need to be part of P, however the restrictions of $\mathcal{E}$ and $\mathcal{M}$ to the morphisms in P form themselves a proper factorization system of P. To simplify notation we also call this restriction a proper $(\mathcal{E}, \mathcal{M})$-factorization system of P. Furthermore C and P are both finitely $\mathcal{M}$-well-powered and both have the joint $\mathcal{M}$-embedding property:

[9] Reminder, a subcategory is full if for all objects a and b that are in P, all morphisms between them in C are also part of P.

5.43 Definition. A locally small category C with an $(\mathcal{E}, \mathcal{M})$-factorization system has the joint $\mathcal{M}$-embedding property if for every two objects a and b, there is an object c such that there are $\mathcal{M}$-subobjects $f : a \to c$ and $g : b \to c$.

Note that the resulting objects for the joint $\mathcal{M}$-embedding property of P also have to be contained in P.

The $\mathcal{M}$-well-poweredness now allows us to define motif parameters φ parallel to Definition 5.10 as linear combinations of $\mathcal{M}$-subobject counting functions $\#\mathrm{Sub}_{\mathcal{M}}(c \to \star)$ for fixed objects c in C. Note that whenever $c \cong c'$, then clearly $\#\mathrm{Sub}_{\mathcal{M}}(c \to \star) = \#\mathrm{Sub}_{\mathcal{M}}(c' \to \star)$, so the different patterns in the linear combination are again always assumed to be pairwise non-isomorphic. We call a pattern P-pure if they are in P and consequently we call φ P-pure if all its patterns are P-pure. As usual Eval_φ denotes the counting function lifting φ to the corresponding promise type-2 function.

As usual, for some subclass D of objects from C that is closed under isomorphisms, we say that a motif parameter φ is *D-good* if all its coefficients are nonnegative integers

and all its patterns are from D. If any such coefficient is negative it is instead called *D-bad*. If the domain D is clear from the context, we will simply call them *good* and *bad* respectively.

Before we get into the proof of our separation, we want to explain why we are restricting our motif parameters to be linear combinations. If there are only finitely many morphisms in $\mathcal{E}$ out of every object, then every monomial $\prod_{i=1}^{k} \#\mathrm{Sub}_{\mathcal{M}}(a_i \to t)$ can be written as a linear combination with non-negative integer coefficients. By extension, every polynomial over $\mathcal{M}$-subobject counting functions can be written as a linear combination. We prove the statement for monomials by giving a characterization as sets, the wanted linearization then follows by taking the cardinality of all sets involved.

5.44 Lemma. *Let C be a locally small category with finite coproducts and an $(\mathcal{E}, \mathcal{M})$-factorization system. Then, assuming the axiom of choice, for any objects $a_1, \dots, a_k, t$ from C we have*

$$\bigtimes_{i=1}^{k} \mathrm{Sub}_{\mathcal{M}}(a_i \to t) \cong \bigsqcup_{b} \mathrm{Sub}_{\mathcal{M}}(b \to t) \times \{(h_1, \dots, h_i) \in \bigtimes_{i=1}^{k} \mathrm{Sub}_{\mathcal{M}}(a_i \to b) \mid \bigsqcup_{i=1}^{k} h_i \in \mathcal{E}\}$$

where b ranges over all non-isomorphic objects in C.

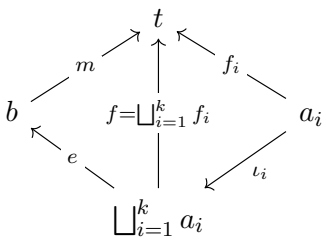
Proof. For each isomorphism class pick one designated object b and for each $\mathcal{M}$-subobject of b in t, choose one designated morphism from $\mathcal{M}$, both using the axiom of choice.

We now directly give the bijection

$$\phi : \bigtimes_{i=1}^{k} \mathrm{Sub}_{\mathcal{M}}(a_i \to t) \to \bigsqcup_{b} \mathrm{Sub}_{\mathcal{M}}(b \to t) \times \{(h_1, \dots, h_i) \in \bigtimes_{i=1}^{k} \mathrm{Sub}_{\mathcal{M}}(a_i \to b) \mid \bigsqcup_{i=1}^{k} h_i \in \mathcal{E}\}\,.$$

Given $\mathcal{M}$-subobjects $f_1, \dots, f_k$ with $f_i \in \mathrm{Sub}_{\mathcal{M}}(a_i \to t)$ for each i, construct $f = \bigsqcup_{i=1}^{k} f_i$. Using the $(\mathcal{E}, \mathcal{M})$-factorization, factorize f through some object b via $e : \bigsqcup_{i=1}^{k} a_k \to b$ from $\mathcal{E}$ and $m : b \to t$ from $\mathcal{M}$. W.l.o.g.[10] we can assume both b and m to be the corresponding designated object and morphism chosen by the axiom of choice. Since the factorization is unique up to isomorphism, this makes (b, m) unique. This leads to the following commutative diagram for each i:

[10] Remember, $\mathcal{E}$ and $\mathcal{M}$ are both closed under composition and contain all isomorphisms. Further, the factorization is unique up to isomorphisms.



Since f_i and m are both in $\mathcal{M}$, this implies that $h_i := e\iota_i$ is in $\mathcal{M}$ by the cancellation property of a factorization system. Due to $e = \bigsqcup_{i=1}^{k} h_i$, we see that $\bigsqcup_{i=1}^{k} h_i$ is in $\mathcal{E}$. Our bijection thus maps $(f_1, \ldots, f_k)$ to $(m, (h_1, \ldots, h_k))$. Note that this is well-defined: if there are morphisms $f'_1, \ldots, f'_k$ from $\mathcal{M}$ and isomorphism $g_1, \ldots, g_k$ with $f'_i = f_i g_i$, then we get $h'_i = h_i g_i$ and thus output the same $\mathcal{M}$-subobjects.

To define Φ^{-1}, let b be some designated object in C, let m be the designated morphism of some $\mathcal{M}$-subobject of b in t and let $(h_1, \ldots, h_k) \in \bigtimes_{i=1}^{k} \mathrm{Sub}_{\mathcal{M}}(a_i \to b)$ with $\bigsqcup_{i=1}^{k} h_i \in \mathcal{E}$. Since $\mathcal{M}$ is closed under composition, we see that each of the mh_i is in $\mathcal{M}$. The bijection then outputs $(mh_1, \ldots, mh_k)$. This function is well-defined by the same argument as above.

Bijectivity now follows from the fact that $\bigsqcup_{i=1}^{k} f_i = m \bigsqcup_{i=1}^{k} h_i$. □

Even more, we can even prove that the coefficients in the linear combination are non-negative integers if the polynomial, given in the binomial basis, has non-negative integer coefficients (potentially leading to negative coefficients in front of the polynomial in monomial basis). For example $\varphi(t) = \binom{\#\mathrm{Sub}_{\mathcal{M}}(a \to t)}{2} = \frac{(\#\mathrm{Sub}_{\mathcal{M}}(a \to t))^2}{2} - \frac{\#\mathrm{Sub}_{\mathcal{M}}(a \to t)}{2}$ can be written as a linear combination of $\mathcal{M}$-subobject counting functions with non-negative integer coefficients.

5.45 Corollary. *Let C be a locally small category with finite coproducts and a proper $(\mathcal{E}, \mathcal{M})$-factorization system. Then, assuming the axiom of choice, for any objects a and t from C we have*

$$\binom{\mathrm{Sub}_{\mathcal{M}}(a \to t)}{k} \cong \bigsqcup_{b} \mathrm{Sub}_{\mathcal{M}}(b \to t) \times \left\{ (h_1, \ldots, h_i) \in \binom{\mathrm{Sub}_{\mathcal{M}}(a \to b)}{k} \mid \bigsqcup_{i=1}^{k} h_i \in \mathcal{E} \right\}$$

where b ranges over all non-isomorphic objects in C.

Proof. In the proof of Lemma 5.44 in the construction of the bijection, whenever $\mathcal{M}$ only contains monos, from $f_i = f_j$, we automatically obtain $h_i = h_j$ and vice-versa. Restricting to only the cases where the morphisms are pairwise disjoint gives the wanted bijection. □

We now continue on to the proof of our separation by first generalizing the notion of set-instantiators:

5.46 Definition (Set-instantiator against (M, c, φ))**.** Let C and P be locally small, finitely $\mathcal{M}$-well-powered categories with a compatible $(\mathcal{E}, \mathcal{M})$-factorization system and where P is a full subcategory of C that is closed under isomorphism.

Let M be a nondeterministic Turing machine, let $c \in C$ and let φ be a P-pure motif parameter.

Let $\top$ be a symbolic top element above the $\mathcal{M}$-subobject poset $\mathcal{P}(c)$, i.e., $\top \not\prec f$ for all $\mathcal{M}$-subobjects $f \in \mathcal{P}(c)$. A *set-instantiator* $\mathcal{SI}$ is a triple of

- some $j \in \mathbb{N}$,
- an instantiation function $\mathsf{inst}_{\mathcal{SI}} : \mathcal{P}(c) \to C_{=j}$, and
- a perception function $\mathsf{perc}_{\mathcal{SI}} : \{0,1\}^* \to \mathcal{P}(c) \cup \{\top\}$,

such that the following property holds for all $\mathcal{M}$-subobjects $f \in \mathcal{P}(c)$.

- $\tau \in \{0,1\}^*$ is an accepting path for the computation $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(f)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \prec f$, and
- $\varphi(\mathsf{inst}_{\mathcal{SI}}(f))) = \varphi(\mathrm{dom}\, f)$.

Again, we think of the perception of an accepting computation path of M to be the $\mathcal{M}$-subobject of c that is observed by M through the oracle queries, while computation paths that do not accept are given perception $\top$. Unfortunately there is no generalized construction for set-instantiators as this has to heavily depend on the actual encoding of the objects as oracle layers. For example consider an encoding where every object is encoded as an oracle layer containing exactly one 1. Then M can determine all information about the object by guessing the position of that single 1 and then querying the oracle once. However, this encoding is not particularly natural and set-instantiators often do exist when encoded naturally. See Sections 5.8.4 and 5.8.5 for two more set-instantiators.

Again, we have the problem that for $\mathcal{M}$-subobjects f and g with $\mathrm{dom}\, f \cong \mathrm{dom}\, g$ we could have $\#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(f)}}(j) \neq \#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(g)}}(j)$, which we do not want to happen in a good function. We thus want to construct an instantiation function that no longer has this problem. We achieve this by creating a very large set-instantiator and then restricting ourselves to some small part of the set-instantiator where we can enforce this property.

Ensuring this structure is achieved by requiring C to have the Ramsey property, relative to $\mathcal{M}$-subobjects.

5.47 Definition. A locally small category C with a $(\mathcal{E}, \mathcal{M})$-factorization system is called $\mathcal{M}$-Ramsey, if for every $a, b \in C$ and $t \in \mathbb{N}$, there is some $c \in C$, such that for every $\Phi : \mathrm{Sub}_{\mathcal{M}}(a \to c) \to \{0, \ldots, t\}$, there is a $f_\Phi \in \mathrm{Sub}_{\mathcal{M}}(b \to c)$ with the property that Φ is constant when restricted to inputs from $\{g \in \mathrm{Sub}_{\mathcal{M}}(a \to c) : g \prec f_\Phi\}$.

See [GLR72] for some categorical conditions that imply the Ramsey property.

We extend this to colorings of all $\mathcal{M}$-subobjects of b, not just those with their domain being a:

5.48 Proposition (Ramsey Theorem)**.** *Let C be an $\mathcal{M}$-Ramsey, locally small and finitely $\mathcal{M}$-well-powered category with a $(\mathcal{E}, \mathcal{M})$-factorization system.*

Let $b \in C$ and let $t \in \mathbb{N}$ be fixed. Then there is a $c \in C$, such that for every $\Phi : \mathcal{P}(c) \to \{0, \dots, t\}$ there is a $f_\Phi \in \mathrm{Sub}_{\mathcal{M}}(b \to c)$ with the property, if $g, h \prec f_\Phi$ with $\mathrm{dom}\, g \cong \mathrm{dom}\, h$, then $\Phi(g) = \Phi(h)$.

Proof. The $\mathcal{P}(b)$ is finite due to C being finitely $\mathcal{M}$-well-powered. Now inductively apply the property that C is $\mathcal{M}$-Ramsey on $\mathrm{dom}\, \mathcal{P}(b)$. □

This proposition can now be used to ensure that for every polynomial-time NTM there must be some small[11] set of inputs where the NTM behaves like a good function. The instantiation function for the inputs where the NTM behaves like a good function now doesn't instantiate individual subobjects anymore, but instead only instantiates objects (the domains of the subobjects). This change is natural since the computed function is now locally a motif parameter and thus invariant under isomorphisms, i.e. the precise subobject doesn't matter, only the isomorphism type of its domain.

[11] Small in the colloquial sense of small, meaning some "small" finite set, not in the sense of category theory where small simply means a set.

5.49 Lemma. *Let C and P be locally small and finitely $\mathcal{M}$-well-powered categories with a compatible $(\mathcal{E}, \mathcal{M})$-factorization system, where P is a full subcategory of C that is closed under isomorphisms and where C is $\mathcal{M}$-Ramsey.*

Let φ be a P-pure motif parameter, let M be any nondeterministic polynomial-time Turing machine computing Eval_φ and let $b \in C$. If for every $c \in C$ there are set-instantiators against (M, c, φ), then there is some $j \in \mathbb{N}$, a function $\mathsf{inst} : \mathrm{dom}\, \mathcal{P}(b) \to C_{=j}$ and a $\mathrm{dom}\, \mathcal{P}(b)$-good function Ψ with $\#acc_{M^{\mathsf{inst}(a)}}(j) = \Psi(a)$ and $\varphi(\mathsf{inst}(a)) = \varphi(a)$ for every $a \in \mathrm{dom}\, \mathcal{P}(b)$.

Proof. We invoke Proposition 5.48 for b and $t := \max\{\varphi(a) : a \in \mathrm{dom}\, \mathcal{P}(b)\}$ to obtain $c \in C$. We now want to color $\mathcal{P}(c)$ according to the behaviour of M on the elements of it, in particular how many accepting paths query each specific $\mathcal{M}$-subobject of c. Before we can do this we have to first embed all potential $\mathcal{M}$-subobjects in $\mathcal{P}(c)$ as oracles. In order to do this we construct a set-instantiator $\mathcal{SI}$ against (M, c, φ) to obtain a $j \in \mathbb{N}$, $\mathsf{inst}_{\mathcal{SI}}$ and $\mathsf{perc}_{\mathcal{SI}}$ as in Definition 5.46.

For $g \in \mathcal{P}(c)$ define $\Phi(g) := \#\mathrm{acc}_{M^{\mathsf{inst}_{\mathcal{SI}}(g)}}(j)$. Note that

$$\Phi(g) \;=\; \left|\left\{\tau \in \{0,1\}^* : \mathsf{perc}_{\mathcal{SI}}(\tau) \prec g\right\}\right| = \sum_{h \prec g} \left|\left\{\tau \in \{0,1\}^* : \mathsf{perc}_{\mathcal{SI}}(\tau) = h\right\}\right|. \quad (5.50)$$

Using Φ as a coloring for Proposition 5.48, we obtain f_Φ with $\mathrm{dom}\, f_\Phi = b$. Note that $\Phi(g)$ now only depends on the isomorphism type of the domain of g for all $g \prec f_\Phi$. By induction we see that the number

$$\left|\left\{\tau \in \{0,1\}^* : \mathsf{perc}_{\mathcal{SI}}(\tau) = g\right\}\right|$$

also depends only on the isomorphism type of $\operatorname{dom} g$, whenever $g \prec f_\Phi$. Denote this number by $\alpha_{\operatorname{dom} g}$ and set $\Psi(\operatorname{dom} g) := \Phi(g)$. The function Φ is well-defined and invariant under isomorphisms of $\operatorname{dom} g$ only for $g \prec f_\Phi$, thus we consider the domain of Ψ to be $\{\operatorname{dom} g \,:\, g \prec f_\Phi\} = \operatorname{dom} \mathcal{P}(b)$. This means that (5.50) simplifies:

$$\Psi(\operatorname{dom} g) \;=\; \sum_{\operatorname{dom} h} \#\mathrm{Sub}_{\mathcal{M}}(\operatorname{dom} h \to \operatorname{dom} g)\alpha_{\operatorname{dom} h} \tag{5.51}$$

where the sum is over all non-isomorphic domains $\operatorname{dom} h$ for $h \in \mathcal{P}(b)$. This implies that Φ is $\operatorname{dom} \mathcal{P}(b)$-good.

Remains to define the function $\mathsf{inst} : \operatorname{dom} \mathcal{P}(b) \to C_{=j}$. Let $a \in \operatorname{dom} \mathcal{P}(b)$, then $\mathsf{inst}(a) := \mathsf{inst}_{\mathcal{SI}}(g_\Phi)$ for some $g_\Phi \prec f_\Phi$ with $\operatorname{dom} g_\Phi \cong a$. The property $\varphi(\mathsf{inst}(a)) = \varphi(a)$ now directly follows from $\mathcal{SI}$ being a set-instantiator. □

Note that Lemma 5.49 only guarantees the local behaviour of M to be $\operatorname{dom} \mathcal{P}(b)$-good. This leaves two possibilities for contradictions: Either the local behaviour is even $(\operatorname{dom} \mathcal{P}(b) \cap P)$-good, or the function grows too quickly for non P-pure inputs. This next Witness Theorem shows that in the first case we can always find an object that proves that M does not compute the correct function using a simple linear algebra argument.

5.52 Theorem (The Witness Theorem)**.** *Let P be a locally small, finitely $\mathcal{M}$-well-powered category with a proper $(\mathcal{E}, \mathcal{M})$-factorization system and the joint $\mathcal{M}$-embedding property.*

Fix a bad $\varphi : P \to \mathbb{N}$. Then there exists $b \in P$ such that for every $(\operatorname{dom} \mathcal{P}(b) \cap P)$-good function $\Psi : (\operatorname{dom} \mathcal{P}(b) \cap P) \to \mathbb{N}$, there exists $w \in (\operatorname{dom} \mathcal{P}(b) \cap P)$ with $\Psi(w) \neq \varphi(w)$.

Proof. Use the joint embedding property of P on $\operatorname{supp}(\varphi)$ to obtain an object $b \in P$, such that there are morphisms $a \to b$ in $\mathcal{M}$ for all $a \in \operatorname{supp}(\varphi)$. We now consider $P' := \operatorname{dom} \mathcal{P}(b) \cap P$, keeping only one representative per isomorphism class. We now prove that the functions $\#\mathrm{Sub}_{\mathcal{M}}(a \to \star)$ for $a \in P'$ are linearly independent: Consider the evaluation matrix A with rows and columns indexed by objects from P', sorted by a total extension of the poset $\mathcal{P}(b)$ and values $A_{ij} := \#\mathrm{Sub}_{\mathcal{M}}(i \to j)$. This is well-defined by Lemma 5.42: if there are morphisms from $\mathcal{M}$, $f : a \to a'$ and $g : a' \to a$, then a and a' are isomorphic and thus only one of them is part of P'. If this matrix is invertible this directly implies that the $\#\mathrm{Sub}_{\mathcal{M}}(a \to \star)$ (i.e. the rows) are linearly independent.

We now prove that A is an upper-diagonal matrix with a 1s on the diagonal and thus invertible. The matrix A is upper-diagonal since we've ordered the rows and columns accordingly, combined with Lemma 5.42. Remaining are the 1s on the diagonal: For any object a, the identity morphisms id_a is contained in $\mathcal{M}$ and thus there is at least one $\mathcal{M}$-subobject in $\mathrm{Sub}_{\mathcal{M}}(a \to a)$. Furthermore every morphism $f : a \to a$ in $\mathcal{M}$ is an isomorphism by Lemma 5.42 and thus the same $\mathcal{M}$-subobject as id_a.

We finish the proof of the Witness Theorem by noticing that the linear independence implies that the patterns and coefficients of any function Ψ with support $\operatorname{supp}(\Psi) \subseteq P'$ are uniquely determined by the evaluations Ψ on P'. Thus there exists a witness $a \in P'$ with $\Psi(a) \neq \varphi(a)$, since Ψ is good while φ is bad (thus, in particular, Ψ and φ are not the same function). □

We need one last condition on C and P: A procedure that, given some c and some $k \in \mathbb{N}$ allows to construct a "bigger" object c' that cannot be distinguished by any P-pure motif parameters, but every non P-pure basis function with patterns from $\operatorname{dom}\mathcal{P}(c)$ evaluates to a value bigger than k. Formally:

5.53 Definition. Let C and P be locally small and finitely $\mathcal{M}$-well-powered categories with a compatible $(\mathcal{E}, \mathcal{M})$-factorization system, where P is a full subcategory of C that is closed under isomorphisms.

We say that C and P fulfill the *blowup property*, if for every object c in C and $k \in \mathbb{N}$ there is some c' in C, such that

$$\begin{aligned} &\#\mathrm{Sub}_{\mathcal{M}}(a \to c') > k && \text{for every } a \in \operatorname{dom}\mathcal{P}(c) \setminus P \\ &\#\mathrm{Sub}_{\mathcal{M}}(a \to c') = \#\mathrm{Sub}_{\mathcal{M}}(a \to c) && \text{for every } a \in P \end{aligned}$$

In the case of graphs, this was achieved by adding isolated vertices.

With this last tool we are finally able to lead the existence of polynomial-time NTMs computing bad ordered graph motif parameters to a contradiction.

5.54 Lemma. *Let C and P be two categories with the following properties:*

- *P is a full subcategory of C, closed under isomorphism.*
- *C and P have a compatible proper $(\mathcal{E}, \mathcal{M})$-factorization system.*
- *C and P are both locally small.*
- *C and P are both finitely $\mathcal{M}$-well-powered.*
- *C and P both have the joint $\mathcal{M}$-embedding property.*
- *C is $\mathcal{M}$-Ramsey.*
- *C and P fulfill the blowup property.*

Let M be any nondeterministic polynomial-time Turing machine computing Eval_φ for some bad P-pure motif parameter φ. If for every $c \in C$ there are set-instantiators against (M, c, φ), then there is a $j \in \mathbb{N}$ and $c \in C_{=j}$ such that $\#acc_{M^c}(j) \neq \varphi(c)$.

Proof. We invoke the Witness Theorem 5.52 with our φ to obtain a $c_1 \in P$ as in the theorem. Using the blowup property, construct c_2 such that

$$\#\mathrm{Sub}_{\mathcal{M}}(a \to c_2) > \max\left\{\varphi(b) \,:\, b \in \operatorname{dom}\mathcal{P}(c_1) \cap P\right\} \quad \text{for every } a \in \operatorname{dom}\mathcal{P}(c_1) \setminus P \tag{5.55}$$

$$\#\mathrm{Sub}_{\mathcal{M}}(a \to c_2) = \#\mathrm{Sub}_{\mathcal{M}}(a \to c_1) \quad \text{for every } a \in P\,. \tag{5.56}$$

We now use Lemma 5.49 to obtain a $j \in \mathbb{N}$, a function $\mathsf{inst} : \operatorname{dom}\mathcal{P}(c_2) \to C_{=j}$ and a $\operatorname{dom}\mathcal{P}(c_2)$-good function Ψ with $\#\mathrm{acc}_{M^{\mathsf{inst}(a)}}(j) = \Psi(a)$ and $\varphi(\mathsf{inst}(a)) = \varphi(a)$ for every $a \in \operatorname{dom}\mathcal{P}(c_2)$.

There are now two possibilities to find the required counterexample w: If there are no basis functions present in Ψ with a pattern in $\operatorname{dom}\mathcal{P}(c_1) \setminus P$, we can invoke the Witness Theorem again, however if any of these basis functions are present we can construct a direct counterexample.

We first handle the case where such a basis function is present, so let the coefficient of $\#\mathrm{Sub}_{\mathcal{M}}(a \to \star)$ in Ψ be positive for some $a \in \operatorname{dom}\mathcal{P}(c_1) \setminus P$. Then

$$\Psi(c_2) \geq \#\mathrm{Sub}_{\mathcal{M}}(a \to c_2) > \max\left\{\varphi(b) \,:\, b \in \operatorname{dom}\mathcal{P}(c_1) \cap P\right\} \geq \varphi(c_1) = \varphi(c_2)$$

by (5.55) and (5.56), combined with the fact that φ is P-pure and thus only has patterns from P. Our counterexample in this case is $w := c_2$.

Otherwise assume that no such basis function is present. Then $\tilde{\Psi}$ obtained from Ψ by restricting to $\operatorname{dom}\mathcal{P}(c_1)$ (i.e. setting all coefficients of $\operatorname{dom}\mathcal{P}(c_2) \setminus \operatorname{dom}\mathcal{P}(c_1)$ to zero), is $(\operatorname{dom}\mathcal{P}(c_1) \cap P)$-good, which is a requirement for the second part of the Witness Theorem 5.52 (we already used it to obtain c_1). Furthermore for the restriction to $\operatorname{dom}\mathcal{P}(c_1)$ we have $\tilde{\Psi}_{|\operatorname{dom}\mathcal{P}(c_1)} = \Psi_{|\operatorname{dom}\mathcal{P}(c_1)}$. We invoke the second part of the Witness Theorem 5.52 to find a point $w \in \operatorname{dom}\mathcal{P}(c_1) \cap P$ with $\Psi(w) = \tilde{\Psi}(w) \neq \varphi(w)$ which is our counterexample.

Independent of how we have obtained our w we observe

$$\#\mathrm{acc}_{M^{\mathsf{inst}(w)}}(j) = \Psi(w) \neq \varphi(w), \tag{5.57}$$

which completes the proof. □

Diagonalizing over all polynomial-time NTMs computing Eval_φ now leads to our general hardness result:

5.58 Theorem. *Let C and P be as in Lemma 5.54 and let φ be a P-pure motif parameter. If set-instatiators against (M, c, φ) exist for all nondeterministic polynomial-time Turing machines M computing Eval_φ and objects $c \in C$, then $\mathrm{Eval}_\varphi \notin \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ unless φ is good.*

Proof. There are two possibilities for φ to not be good: Either φ is bad or φ has a non-integer coefficient.

We first treat the case where φ is bad. Assume for the sake of contradiction $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$. Then there is a polynomial-time NTM M computing Eval_φ. Due to the existence of set-instantiators we can invoke Lemma 5.54 and obtain $j \in \mathbb{N}$ and $w \in C_{=j}$ with $\#\mathrm{acc}_{M^w}(j) \neq \varphi(w)$. This however, means that M in fact does not compute Eval_φ, a contradiction.

Second we treat the case where φ has a non-integer coefficient. Similarly to the proof of Theorem 5.52, using the joint $\mathcal{M}$-embedding property of C on $\mathrm{supp}(\varphi)$ to obtain an object $b \in C$, such that there are morphisms $a \to b$ in $\mathcal{M}$ for all $a \in \mathrm{supp}(\varphi)$. Using the $\prec$ order, find some minimal element f in $\mathcal{P}(b)$, such that the pattern $\mathrm{dom}\, f$ has a non-integer coefficient α. Consider $\varphi(\mathrm{dom}\, f)$. Recall the evaluation matrix A from the proof of Theorem 5.52. Since it is upper-diagonal with 1s on the diagonal, we see that the only patterns affecting $\varphi(\mathrm{dom}\, f)$ are the ones with patterns $\mathrm{dom}\, g$, for $g \prec f$. All of them have integer coefficients and thus $\varphi(\mathrm{dom}\, f) = n + \alpha \cdot \#\mathrm{Sub}_{\mathcal{M}}(\mathrm{dom}\, f \to \mathrm{dom}\, f)$ for some $n \in \mathbb{Z}$. The 1s on the diagonal correspond to $\#\mathrm{Sub}_{\mathcal{M}}(\mathrm{dom}\, f \to \mathrm{dom}\, f) = 1$, so $\varphi(\mathrm{dom}\, f) = n + \alpha \notin \mathbb{Z}$. We conclude, φ does not only output integers and thus $\mathrm{Eval}_\varphi \notin \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$. □

In our general theorem above, we do not get a dichotomy, since we do not need to have an upper bound. It might be the case that the evaluation of the motif parameter is not contained in $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ even when the coefficients are all non-negative integers. However, in our applications below it is easy to show that this does not happen.

5.8.3 Ordered Graphs

To show how this generalization compares to the proof of Theorem 5.17, we first show how this categorical generalization can be used to reprove the separation of Theorem 5.17:

We consider $C = \mathbf{FinOrdGraphs}$ to be the category of all finite ordered graphs. The objects of this category are the finite[12] ordered graphs while the morphisms are the monotone graph homomorphisms. The monomorphisms of **FinOrdGraphs** are the injective monotone graph homomorphisms, i.e. (not necessarily strong) monotone graph embeddings. The epimorphism are the vertex surjective monotone graph homomorphisms, i.e. $f : H \to G$ is epi, if for every $v \in V(G)$ there is some $v' \in V(H)$ such that $f(v) = v'$. The category P of pure objects is obtained by restricting C to those ordered graphs without isolated vertices and keeping all morphisms between such ordered graphs, making P a full subcategory of C. Clearly P is closed under isomorphisms, the property of having isolated vertices is preserved by isomorphisms. Both categories are locally finite, there are only finitely many monotone graph homomorphisms between two fixed graphs, and thus locally small.

[12] The finite was implicit throughout this entire paper, but we make it explicit here to distinguish of the category of ordered graphs, which usually includes infinite graphs.

Next we describe the proper $(\mathcal{E}, \mathcal{M})$-factorization system: We choose $\mathcal{E}$ as the epimorphisms and we choose $\mathcal{M}$ as the strong monotone graph embeddings. This makes the $\mathcal{M}$-subobjects of some monotone graph G precisely the induced subgraphs of G (or rather an equivalence class, represented by each of the induced subgraphs). We prove this indeed forms a factorization system:

For that let $f : H \to G$ be an monotone graph homomorphism. Consider $G[f(V(H))]$, the subgraph of G induced by the image of the vertices of H under f. Let $m : G[f(V(H))] \to G$ be the canonical strong monotone graph embedding (the identity) and let $e : H \to G[f(V(H))]$ be obtained from f by restricting its codomain. Then e is surjective on the vertices and thus $e \in \mathcal{E}$ and $m \in \mathcal{M}$ with $f = me$. On the other hand, let $f = m'e'$ be a different factorization with $e' : H \to F$ in $\mathcal{E}$ and $m' : F \to G$ in $\mathcal{M}$ through some ordered graph F. Due to $m' \in \mathcal{M}$, F is isomorphic to some induced subgraph of G, say $F \cong G[X]$ with isomorphism $g : F \to G[X]$ for some $X \subseteq V(G)$. W.l.o.g. assume the strong monotone graph embedding $G[X] \to G$ to be the identity (extended to all of G as a codomain). Then $X = ge'(V(H)) = f(V(H))$ and thus $mge' = me$.

Compatibility of the factorization system between C and P is given by the following fact: Let $f : H \to G$ factor as $f = me$ with $e : H \to F$ in $\mathcal{E}$ and $m : F \to G$ in $\mathcal{M}$ and let $v \in V(F)$ be an isolated vertex in F. Then there is a $v' \in V(H)$ with $e(v') = v$ due to vertex surjectivity of e. The images of all neighbors of v' in H under e are neighbors of v in F, due to e being a homomorphism. Thus v' has to be an isolated vertex in H. Together this implies that F is pure if H is pure.

There are only finitely many induced subgraphs for any ordered graph G, making C and P both $\mathcal{M}$-well-powered. The joint $\mathcal{M}$-embedding property for ordered graphs G, H is given by the disjoint union of G and H with the obvious inlusions. The category C being $\mathcal{M}$-Ramsey is given by Lemma 5.22. The blowup property is fulfilled by adding isolated vertices to the ordered graph we are trying to blow up.

The sets $C_{=j}$ are precisely the ordered graphs G with $V(G) = \{1, \dots, j\}$, represented as oracles as before. The corresponding set-instantiators have been constructed in Lemma 5.21.

Theorem 5.58 then reproves the separation of Theorem 5.17.

5.8.4 Finite Vector Spaces over Finite Fields

Our first new example is the category $C := \mathbf{FinVect}_{\mathbb{F}_p}$ of finite vector spaces over a fixed finite field $\mathbb{F}_p$. Here, an NTM is given the size p^d of an ambient vector space $\mathbb{F}_p^d$ and some vector space $V \subseteq \mathbb{F}_p^d$. It can query V by querying the oracle for individual vectors, i.e. it can ask whether $(v_1, \dots, v_d) \in V$ for $v_1, \dots, v_d \in \mathbb{F}_p$. The vector $(v_1, \dots, v_d)$ is suitably encoded as a binary string of some fixed length bitsize(d), only depending on p and d, and thus V itself can be encoded as a subset of $\{0,1\}^{\text{bitsize}(d)}$. Our sets $C_{=p^d}$ are then precisely all the vector spaces $V \subseteq \mathbb{F}_p^d$ and bitsize(d) is poly-logarithmic in p^d.

Due to the ambient space we can already pad a vector space by embedding it into a larger ambient space. This allows us to choose P and C to be the same category $\mathbf{FinVect}_{\mathbb{F}_p}$ that we now describe: Let $\mathbb{F}_p$ be some fixed finite field. The objects of $\mathbf{FinVect}_{\mathbb{F}_p}$ are all the finite vector spaces over $\mathbb{F}_p$ and the morphisms between them are the vector space homomorphisms. This makes the epimorphisms the surjective homomorphisms, while the monomorphisms are the injective homomorphisms. For some vector spaces $V, W \in \mathbf{FinVect}_{\mathbb{F}_p}$, the homomorphisms from V to W can be represented by the $\dim W \times \dim V$ matrices over $\mathbb{F}_p$. We thus conclude that $\mathbf{FinVect}_{\mathbb{F}_p}$ is locally small and even locally finite. The proper $(\mathcal{E}, \mathcal{M})$ factorization system is given by choosing $\mathcal{E}$ to be all epimorphisms and $\mathcal{M}$ to be all monomorphisms. That this indeed forms a factorization system follows from $\mathbf{FinVect}_{\mathbb{F}_p}$ being an abelian category, we refer the interested reader to [Mac98][Chapter VIII, Section 3, Proposition 1] for the details. The $\mathcal{M}$-subobjects of some vector space V are represented precisely by the subspaces $W \subseteq V$, making $\mathbf{FinVect}_{\mathbb{F}_p}$ finitely $\mathcal{M}$-well-powered. We thus have $\#\mathrm{Sub}_{\mathcal{M}}(W \to V) = \binom{\dim(V)}{\dim(W)}_p$, the Gaussian binomial coefficient (see [KC02][Theorem 7.1] for details on this). As such the motif parameters $\varphi : \mathbf{FinVect}_{\mathbb{F}_p} \to \mathbb{Q}$ simply depend on the dimension of the input. Furthermore $P = C$ implies that the P-pure motif parameters are precisely the motif parameters. We will thus only deal with motif parameters in the following. The category $\mathbf{FinVect}_{\mathbb{F}_p}$ is well known to admit finite coproducts: the direct sum of vector spaces. The coproduct injections for coproducts in $\mathbf{FinVect}_{\mathbb{F}_p}$ are always mono, implying the joint $\mathcal{M}$-embedding property. The Graham-Leeb-Rothschild Theorem [GLR72] states that $\mathbf{FinVect}_{\mathbb{F}_p}$ is $\mathcal{M}$-Ramsey. The blowup property for C and P is vacuously true due to $P = C$.

The only thing left to apply Theorem 5.58 is to prove the existence of set-instantiators:

5.59 Lemma. *Let $\varphi : \mathbf{FinVect}_{\mathbb{F}_p} \to \mathbb{N}$ be a motif parameter, let M be a polynomial-time NTM computing Eval_φ and let $V \in \mathbf{FinVect}_{\mathbb{F}_p}$. Then there is a set-instantiator against (M, V, φ).*

Proof. For now let the dimension d of the ambient space be undetermined, we will choose it accordingly later. Consider uniformly chosen vector space homomorphisms $h : V \to \mathbb{F}_p^d$. Our goal is to use the union bound to show that there is a choice for h such that:

1. h is injective.
2. For each subspace $W \subseteq V$, all accepting paths of the computation $\#\mathrm{acc}_{M^{h(W)}}(p^d)$ do not query the oracle for any vectors in $h(V \setminus W)$.

If each of these events (we count property 2 individually for each of the finitely many subspaces) happens with high probability with growing d, then eventually there is an h fulfilling all of them.

For now, fix some subspace $W \subseteq V$. Further group the homomorphisms, by their image of W. Call one such group H. Then the set S_H of all vectors queried by all accepting

paths of the computation $\#\mathrm{acc}_{M^{h(W)}}(p^d)$ is the same for all $h \in H$. Choose a basis $w_1, \ldots, w_n$ of W and extend it with vectors $v_1, \ldots, v_m$ to a basis of V. The images $h(v_1), \ldots, h(v_m)$ are now independently and uniformly distributed in $\mathbb{F}_p^d$. Any fixed vector $v \in V \setminus W$ can be written as $\sum_{i=1}^{n} \alpha_i w_i + \sum_{i=1}^{m} \beta_i v_i$ with some $\beta_i \neq 0$ and thus $h(v)$ is also uniformly distributed in $\mathbb{F}_p^d$. We thus have

$$\Pr_{h \in H}[h(v) \in S_H] = \frac{|S_H|}{p^d}.$$

Combining this using the union bound and $|V \setminus W| \leq |V|$ we get

$$\Pr_{h \in H}[\exists v \in V \setminus W \text{ with } h(v) \in S_H] \leq |V| \cdot \frac{|S_H|}{p^d}.$$

Remains to estimate $|S_H|$. Let $t_M(p^d)$ be the worst-case running-time of M on input p^d. Each oracle query can query exactly one vector, there are at most $\varphi(h(W))$ many accepting paths and each accepting path can query the oracle at most $t_M(p^d)$ many times. This gives the bound $|S_H| \leq \varphi(h(W)) \cdot t_M(p^d)$ on the size of S_H. This bound is polynomial in d: M is polynomial-time and $\varphi(h(W))$ is bounded by the constant $\max_{W' \subseteq V} \varphi(W')$ due to φ being a motif parameter.

Across all homomorphisms $h : V \to \mathbb{F}_p^d$ (not just those from H) we thus get that all accepting paths of the computation $\#\mathrm{acc}_{M^{h(W)}}(p^d)$ do not query the oracle for any vectors in $h(V \setminus W)$ with a probability of at least $1 - |V| \cdot \frac{\mathrm{poly}(d)}{p^d}$.

Remains to show that h is also injective with high probability: Conversely, if h is not injective, then there is some non-zero $v \in V$ with $h(v) = 0$. For a fixed $v \in V$, $h(v)$ is again uniformly distributed, so the chance of this happening for this specific v is $\frac{1}{p^d}$. Using the union bound yet again, we get that the chance that h is injective is at least $1 - |V| \cdot \frac{1}{p^d}$.

Since $|V|$ is constant, a final union bound proves that there is a large enough d, such that there is a homomorphism h fulfilling properties 1 and 2.

We now construct our set-instantiator with $j = p^d$ for the d chosen above. The instantiation function for a subspace $W \subseteq V$ is $\mathsf{inst}_{\mathcal{SI}}(W) = h(W)$. For any computation path $\tau \in \{0,1\}^\star$ of a computation $\#\mathrm{acc}_{M^{h(W)}}(j)$ denote by $S_\tau \subseteq \mathbb{F}_p^d$ the vectors that are queried by the computation. We set

$$\mathsf{perc}_{\mathcal{SI}}(\tau) := \begin{cases} \mathrm{Span}(h^{-1}(S_\tau)) & \text{if } \tau \text{ is an accepting path of the computation } \#\mathrm{acc}_{M^{h(V)}}(j), \\ \top & \text{otherwise} \end{cases}$$

We check the properties of a set-instantiator. Clearly, $\varphi(\mathsf{inst}_{\mathcal{SI}}(W)) = \varphi(W)$ for all subspaces $W \subseteq V$ due to h being injective, making $h(W)$ and W isomorphic. Further, we need that for all subspaces $W \subseteq V$ we have that $\tau \in \{0,1\}^\star$ is an accepting path for the computation $\#\mathrm{acc}_{M^{h(W)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \subseteq W$. For this, fix some subspace $W \subseteq V$ and

let τ be an accepting computation of the computation $\#\mathrm{acc}_{M^{h(W)}}(j)$. By our choice of h we know that no other vertices of $h(V \setminus W)$ are queried, or in other words, $h^{-1}(S_\tau) \subseteq W$ and thus τ is also an accepting path for the computation $\#\mathrm{acc}_{M^{h(V)}}(j)$. This directly implies $\mathsf{perc}_{\mathcal{SI}}(\tau) = \mathrm{Span}(h^{-1}(S_\tau)) \subseteq W$. On the other hand, let $\mathsf{perc}_{\mathcal{SI}}(\tau) \subseteq W$, then $\mathsf{perc}_{\mathcal{SI}}(\tau) \neq \top$ and τ is an accepting path of the computation $\#\mathrm{acc}_{M^{h(V)}}(j)$. We conclude that τ is also an accepting path of the computation $\#\mathrm{acc}_{M^{h(W)}}(j)$ due to $\mathsf{perc}_{\mathcal{SI}}(\tau) = \mathrm{Span}(h^{-1}(S_\tau)) \subseteq W$. □

Applying Theorem 5.58 now proves the main result for $\mathbf{FinVect}_{\mathbb{F}_p}$.

5.60 Theorem. *Let $\varphi : \mathbf{FinVect}_{\mathbb{F}_p} \to \mathbb{N}$ be a motif parameter. Then $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot^{\cdot^{\cdot}}}$ iff φ is good.*

Proof. Theorem 5.58 gives the lower bound.

For the upper bound, since φ is a positive integer linear combination of finitely many basis functions and $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot^{\cdot^{\cdot}}}$ is closed under such linear combinations it suffices to consider $\varphi(V) = \#\mathrm{Sub}_{\mathcal{M}}(W \to V)$ for some vector space W. We thus need to count the number of $\dim(W)$-dimensional subspaces of $V \in \mathbb{F}_p^d$, given p^d and oracle access to V as an input. For this, guess $p^{\dim(W)}$ many vectors in $\mathbb{F}_p^d$ that form a $\dim(W)$-dimensional vector space (sorted in some way, such that the same vector space is only guessable in one way) and query the oracle whether all of them are part of V. If this is the case, accept, otherwise reject. Clearly this counts the number of $\dim(W)$-dimensional subspaces of V and runs in time polynomial in d ($\dim(W)$ is a fixed constant). We conclude $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot^{\cdot^{\cdot}}}$. □

Note that the upper bound in the previous algorithm is only a $\mathsf{Pr}\text{-}\#\mathsf{P}^{\cdot^{\cdot^{\cdot}}}$ upper bound. The NTM M is unable to verify whether the oracle indeed encodes a valid vector space.

5.8.5 Parameter Sets

Our final example are the so called parameter sets. Fix some finite alphabet A. An n-parameter set X is determined by a point $x^0 \in A^N$ and disjoint non-empty sets $\Lambda_1, \dots, \Lambda_n \subseteq \{1, \dots, N\}$. It contains all points $f \in A^N$ with

$$x_i = \begin{cases} x_i^0 & \text{if } i \notin \bigcup_{j=1}^n \Lambda_j, \\ x_{i'} & \text{if } i, i' \in \Lambda_j \text{ for some } j = 1, \dots, n. \end{cases}$$

Note that different choices of x^0 or orderings of the Λ_i can lead to the same parameter set and we do not distinguish between them, two parameter sets are the same if the underlying set is the same. A p-parameter subset $Y \subseteq X$ is simply a p-parameter set Y with $Y \subseteq X$ as sets. One can think of parameter sets as special affine vector spaces.

We choose $C = P$ to be the same category $\mathbf{ParSets}_A$. The category $\mathbf{ParSets}_A$ is the category of all parameter sets over A. A morphism $f : X \to Y$ between parameter sets

$X \subseteq A^N$ and $Y \subseteq A^{N'}$, is now a function between the underlying sets of X and Y, such that for any parameter set $X' \subseteq X$, we have that $f(X') \subseteq Y$ is a parameter set. Such a morphism f is mono if the underlying function is injective and epi if it is surjective. Furthermore f is an isomorphism if the underlying function is an isomorphism. This is only possible if X and Y have the same number of parameters. But also conversely, if X and Y have the same number of parameters, then there is an isomorphism between them. Since the morphisms $f : X \to Y$ are always a subset of the functions $X \to Y$ (as finite sets), we conclude that $\mathbf{ParSets}_A$ is locally small and even locally finite. The proper $(\mathcal{E}, \mathcal{M})$-factorization system is now given by choosing $\mathcal{E}$ to be the set of all epimorphisms and $\mathcal{M}$ to be the set of all monomorphisms. Any morphism $f : X \to Y$ then uniquely factors through $\mathrm{im}(f)$ in a natural way. The $\mathcal{M}$-subobjects of some parameter set X are canonically represented by the parameter sets $Y \subseteq X$, implying that $\mathbf{ParSets}_A$ is finitely $\mathcal{M}$-well-powered.

Furthermore the category has finite coproducts: Remember that there is essentially (i.e. up to isomorphisms) only one parameter set for each number of parameters. The coproduct of "the" n-parameter set A^n and "the" m-parameter set A^m is then "the" $n+m$-parameter set A^{n+m}. The corresponding coproduct injections $f : A^n \to A^{n+m}$ and $g : A^m \to A^{n+m}$ simply map their inputs to either the first n or the last m parameters of A^{n+m}, respectively. Both f and g are mono, implying the joint $\mathcal{M}$-embedding property.

Using the canonical n-parameter set A^n, we see that the number of m-parameter sets $Y \subseteq A^n$ is given as the number of ways to partition $|A| \cup \{1, \dots, n\}$ into $|A| + m$ disjoint non-empty subsets, such that the elements from A are mapped into distinct subsets, w.l.o.g. the first $|A|$ ones. These first $|A|$ subsets represent the parameters replaced by individual constants from A. Thus, for an m-parameter set X and an n-parameter set Y we have

$$\#\mathrm{Sub}_{\mathcal{M}}(X \to Y) = \left\{ \begin{array}{c} |A| + n \\ |A| + m \end{array} \right\}_{|A|},$$

an $|A|$-Stirling number of the second kind (see [Bro84] for details on $|A|$-Stirling numbers). Similarly to the category $\mathbf{FinVect}_{\mathbb{F}_p}$, the motif parameters $\mathbf{ParSets}_A \to \mathbb{Q}$ simply depend on the number of parameters of the input. The Graham-Rothschild Theorem (see [Neš95, Theorem 10.4] for a proof) states that $\mathbf{ParSets}_A$ is $\mathcal{M}$-Ramsey. The blowup property for C and P is vacuously true due to $P = C$.

The way we attach parameter sets as oracles to an NTM is as follows: The NTM is given the size $|A|^N$ of an ambient space A^N and some parameter set $X \subseteq A^N$. It can query X by asking the oracle, whether $(x_1, \dots, x_N) \in X$ for some $x_1, \dots, x_N \in A$. The tuple $(x_1, \dots, x_N)$ is suitably encoded as a binary string of some fixed length $\mathrm{bitsize}(N)$, only depending on N and A, and thus X itself can be encoded as a subset of $\{0,1\}^{\mathrm{bitsize}(N)}$. Our sets $C_{|A|^N}$ are then precisely all the parameter sets $X \subseteq A^N$ and $\mathrm{bitsize}(N)$ is poly-logarithmic in $|A|^N$.

The only thing left to apply Theorem 5.58 is to prove the existence of set-instantiators:

5.61 Lemma. *Let $\varphi : \mathbf{ParSets}_A \to \mathbb{N}$ be a motif parameter, let M be a polynomial-time NTM computing Eval_φ and let $X \in \mathbf{ParSets}_A$. Then there is a set-instantiator against (M, X, φ).*

Proof. Let $X \in \mathbf{ParSets}_A$ be an n-parameter set with sets $\Lambda_1, \dots, \Lambda_n$.

For now let the dimension N of the ambient space be undetermined, we will choose it accordingly later. Fix some $z'^0 \in A^N$. Uniformly at random choose non-empty sets $\Lambda'_1 \subseteq \{1, \dots, \frac{N}{n}\}, \dots, \Lambda'_n \subseteq \{1 + \frac{n-N}{n}, \dots, N\}$. This defines an n-parameter set $Z \subseteq A^N$. This directly defines a bijection h from parameter sets $Y \subseteq X$ to parameter sets $Y' \subseteq Z$: Whenever a parameter from X is replaced by a constant to obtain Y, we replace the same parameter in Z by the same constant and whenever some parameters of X are merged to obtain Y, we merge the same parameters of Z to obtain Y'. This naturally also extends to a bijection between elements $x \in X$ and $z \in Z$ (they are 0-parameter sets). Note that $Z = h(X)$.

As usual we want to use the union bound to show that there is a choice of the Λ'_i (and thus h), such that for every parameter set $Y \subseteq X$, all accepting paths of the computation $\#\mathrm{acc}_{M^{h(Y)}}(|A|^N)$ do not query the oracle for any elements in $h(Z) \setminus h(Y)$.

If this happens with high probability with growing N for all of the finitely many parameter sets $Y \subseteq X$, then eventually such an h exists. For now fix some parameter set $Y \subseteq X$. Further group the possible h by the image of Y. Call one such group H. Then the set S_H of all elements queried by all accepting paths of the computation $\#\mathrm{acc}_{M^{h(Y)}}(|A|^N)$ is the same for all $h \in H$. Let $x \in h(X) \setminus h(Y)$ be fixed. Split x into the natural blocks of $\frac{N}{n}$ coordinates. Then there is at least one block, say the i-th block, where x needs to be changed to be in $h(Y)$. In particular, only one choice of entries in that block works. However, since Λ'_i is uniformly distributed, those entries vary across the h. We conclude

$$\Pr_{h \in H}[x \in S_H] \leq \frac{|S_H|}{2^{\frac{N}{n}}}.$$

Combining this using the union bound and $|h(X) \setminus h(Y)| \leq |h(X)| = |X|$ we get

$$\Pr_{h \in H}[\exists x \in h(X) \setminus h(Y) \text{ with } x \in S_H] \leq |X| \cdot \frac{|S_H|}{2^{\frac{N}{n}}}.$$

It remains to estimate $|S_H|$. Let $t_M(|A|^N)$ be the worst-case running time of M on input $|A|^N$. Each oracle query can query exactly one element, there are at most $\varphi(h(X)) = \varphi(X)$ many accepting paths and each accepting path can query the oracle at most $t_M(|A|^N)$ times. This gives the bound of $|S_H| \leq \varphi(X) \cdot t_M(|A|^N)$. This bound is polynomial in N: M is polynomial-time and $\varphi(X)$ is constant, due to X being fixed. Across all possible functions h (not just those from H), we thus get that all accepting paths of the computation $\#\mathrm{acc}_{M^{h(Y)}}(|A|^N)$ do not query the oracle for elements in $h(X) \setminus h(Y)$ with a probability of at least $1 - |X| \cdot \frac{\mathrm{poly}(N)}{2^{\frac{N}{n}}}$.

Since $|X|$ and n are constant, a final union bound proves that there is some large enough N, such that there is some h that has our required property.

We now construct our set-instantiator with $j = |A|^N$ for the N above. This instantiation function for some parameter set $Y \subseteq X$ is $\mathsf{inst}_{\mathcal{SI}}(Y) = h(Y)$. For any computation path $\tau \in \{0,1\}^\star$ of a computation $\#\mathrm{acc}_{M^{h(Y)}}(|A|^N)$ denote by $S_\tau \subseteq h(X)$ the elements that are queried by the computation. We set

$$\mathsf{perc}_{\mathcal{SI}}(\tau) := \begin{cases} \mathrm{Span}(h^{-1}(S_\tau)) & \text{if } \tau \text{ is an accepting path of the computation } \#\mathrm{acc}_{M^{h(X)}}(j), \\ \top & \text{otherwise} \end{cases}$$

where $\mathrm{Span}(U)$ denotes the parameter set with the lowest amount of parameters containing U. We check the properties of a set-instantiator. Clearly $\varphi(\mathsf{inst}_{\mathcal{SI}}(Y)) = \varphi(Y)$ for all parameter sets $Y \subseteq X$ due to Y and $h(Y)$ having the same number of parameters and thus being isomorphic. Further we need that for all parameter sets $Y \subseteq X$ we have that $\tau \in \{0,1\}^\star$ is an accepting path for the computation $\#\mathrm{acc}_{M^{h(Y)}}(j)$ iff $\mathsf{perc}_{\mathcal{SI}}(\tau) \subseteq Y$. For this, fix some parameter set $Y \subseteq X$ and let τ be an accepting computation of the computation $\#\mathrm{acc}_{M^{h(Y)}}(j)$. By our choice of h we know that no other elements of $h(X) \setminus h(Y)$ are queried, or in other words, $h^{-1}(S_\tau) \subseteq Y$ and thus τ is also an accepting path for the computation $\#\mathrm{acc}_{M^{h(X)}}(j)$. This directly implies $\mathsf{perc}_{\mathcal{SI}}(\tau) = \mathrm{Span}(h^{-1}(S_\tau)) \subseteq Y$. On the other hand, if $\mathsf{perc}_{\mathcal{SI}}(\tau) \subseteq Y$, then $\mathsf{perc}_{\mathcal{SI}}(\tau) \neq \top$ and τ is an accepting path of the computation $\#\mathrm{acc}_{M^{h(X)}}(j)$. We conclude that τ is also an accepting path of the computation $\#\mathrm{acc}_{M^{h(Y)}}(j)$ due to $\mathsf{perc}_{\mathcal{SI}}(\tau) = \mathrm{Span}(h^{-1}(S_\tau)) \subseteq Y$. □

Applying Theorem 5.58 now proves the main result for $\mathbf{ParSets}_A$.

5.62 Theorem. *Let $\varphi : \mathbf{ParSets}_A \to \mathbb{N}$ be a motif parameter. Then $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ iff φ is good.*

Proof. Theorem 5.58 gives the lower bound.

For the upper bound, since φ is a positive integer linear combination of finitely many basis functions and $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ is closed under such linear combinations it suffices to consider $\varphi(X) = \#\mathrm{Sub}_{\mathcal{M}}(Y \to X)$ for some n-parameter set Y. We thus need to count the number of n-parameter sets of $Y \subseteq X \subseteq A^N$, given $|A|^N$ and oracle access to X as an input. For this now guess $|A|^n$ many elements in A^N that form a n-parameter set (sorted in some way, such that the same parameter set is only guessable in one way) and query the oracle whether all of them are part of X. If this is the case, accept, otherwise reject. Clearly this counts the number of n-parameter subsets of X and runs in time polynomial in N (n is a fixed constant). We conclude $\mathrm{Eval}_\varphi \in \mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$. □

Again, the upper bound in the previous algorithm is only a $\mathsf{Pr}\text{-}\#\mathsf{P}^{\circlearrowleft}$ upper bound. The NTM M is unable to verify whether the oracle indeed encodes a valid parameter set.

5.9 Non-negativity of the Example

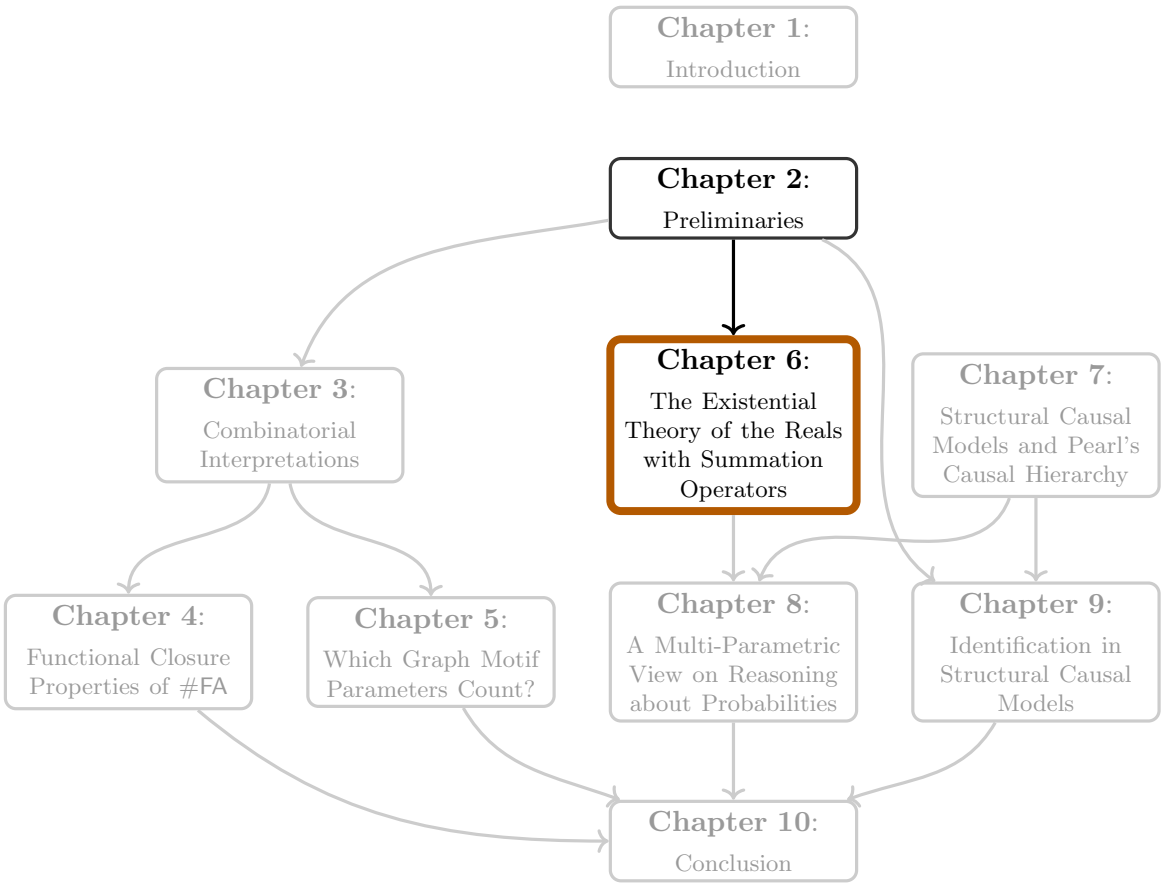
Recall from Example 5.2 the graph motif parameter

$$f(G) = \#\mathrm{Ind}(\ \to G) - \#\mathrm{Ind}(\ \to G) + \#\mathrm{Ind}(\ \to G) \\ + 2\,\#\mathrm{Ind}(\ \to G) + 4\,\#\mathrm{Ind}(\ \to G).$$

We claim that $f(G) \geq 0$ for every G. It is easy to see that $f(G) = \sum_C f(C)$, where the sum is over all connected components C of G. It remains to prove that $f(C) \geq 0$ for each connected component C. This is obvious for connected components with at most 2 vertices. For every connected component C that is a triangle, we have $f(C) = 3 - 1 = 2 \geq 0$. For every other connected component, for each triangle T we choose an induced subgraph $F(T)$ of C that contains T and that is isomorphic to , , or . These (not necessarily unique) choices partition the set of triangles of C into three disjoint sets $\mathcal{T}(\ )$, $\mathcal{T}(\ )$ and $\mathcal{T}(\ )$ with $\#\mathrm{Ind}(\ \to C) = |\mathcal{T}(\ )| + |\mathcal{T}(\ )| + |\mathcal{T}(\ )|$.

For an induced subgraph H of C, if $H \cong$, then there is at most 1 triangle T with $F(T) = H$. If $H \cong$, then there are at most 2 triangles T with $F(T) = H$. If $H \cong$, then there are at most 4 triangles T with $F(T) = H$. Hence,

$$\begin{aligned} f(C) &\geq \#\mathrm{Ind}(\ \to C) + 2\,\#\mathrm{Ind}(\ \to C) + 4\,\#\mathrm{Ind}(\ \to C) - \#\mathrm{Ind}(\ \to C) \\ &= \big(\#\mathrm{Ind}(\ \to C) - |\mathcal{T}(\ )|\big) + \big(2\,\#\mathrm{Ind}(\ \to C) - |\mathcal{T}(\ )|\big) \\ &\quad + \big(4\,\#\mathrm{Ind}(\ \to C) - |\mathcal{T}(\ )|\big) \\ &\geq 0. \end{aligned}$$



The Existential Theory of the Reals with Summation Operators

6.1 Introduction

[1]Remember, we have defined ETR in Section 2.5.5.

This chapter focuses on extending the syntax of ETR-formulas[1] to allow the use of summation operators in addition to the functional symbols + and ·. This research direction, initiated in [ZBL23], was motivated by an attempt to accurately characterize the computational complexity of satisfiability problems for probabilistic and causal reasoning across "Pearl's Causal Hierarchy" (PCH) [SP08; Pea09; BCII22]. We will cover and extend this motivating example in more detail in Chapter 8, while for now we will focus on the complexity class they introduce: succ-$\exists\mathbb{R}$.

The new natural class succ-$\exists\mathbb{R}$ can be viewed as a succinct variant of $\exists\mathbb{R}$. Perhaps, one of the most notable complete problems for the new class is the problem called Σ_{vi}-ETR ("*vi*" stands for variable indexing). It is defined as an extension of ETR by adding to the signature an additional summation operator[2] $\sum_{x_j=0}^{1}$ which can be used to *index* the quantified variables. To this end, the authors define variables of the form $x_{\langle x_{j_1},\ldots,x_{j_m}\rangle}$, which represent indexed variables with the index given by $x_{j_1},\ldots,x_{j_m}$ interpreted as a number in binary. They can only be used when variables $x_{j_1},\ldots,x_{j_m}$ occur in the scope of summation operators with range $\{0,1\}$.

[2]In [ZBL23], the authors assume arbitrary integer lower and upper bound in $\sum_{x_j=a}^{b}$. It is easy to see that, w.l.o.g., one can restrict a and b to binary values.

E.g., $\exists x_0 \ldots \exists x_{2^N-1} \sum_{e_1=0}^{1} \cdots \sum_{e_N=0}^{1} (x_{\langle e_1,\ldots,e_N\rangle})^2 = 1$ is a Σ_{vi}-ETR sentence[3] encoding a unit vector in $\mathbb{R}^{2^N}$. Note that sentences of Σ_{vi}-ETR allow the use of exponentially many variables. Another example sentence is $\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}(x_1+x_2)(x_1+(1-x_2))(1-x_1)=0$ that models the UNSAT instance $(p\vee q)\wedge(p\vee\overline{q})\wedge\overline{p}$. It shows that the summation operator can also be used in Σ_{vi}-ETR formulas in a standard way.

[3]We represent the instances in Σ_{vi}-ETR omitting the (redundant) block of existential quantifiers, so the encoding of the example instance has length polynomial in N.

Analogously to $\exists\mathbb{R}$, succ-$\exists\mathbb{R}$ is an intermediate class between the exponential versions of NP and PSPACE:

$$\mathsf{NP} \subseteq \exists\mathbb{R} \subseteq \mathsf{PSPACE} \subseteq \mathsf{NEXP} \subseteq \text{succ-}\exists\mathbb{R} \subseteq \mathsf{EXPSPACE}. \tag{6.1}$$

An interesting challenge, in view of the new class, is to determine whether it contains harder problems than NEXP and to examine the usefulness of succ-$\exists\mathbb{R}$-completeness as a tool for understanding the apparent intractability of natural problems. A step in these directions, that we take in this chapter, is to express succ-$\exists\mathbb{R}$ in terms of machine models over the reals, which in the case of $\exists\mathbb{R}$ yield an elegant and useful characterization by $\mathsf{NP}_{\text{real}}$ [EHM24].

Additionally we study the different restrictions on which the summation operators in ETR are allowed to be used and the computational complexity of deciding the resulting problems. In particular, we investigate $\exists\mathbb{R}^{\Sigma}$ – the class based on ETR enriched with a standard summation operator (without variable indexing), and succ-$\exists\mathbb{R}_{\text{poly}}$ which is based on Σ_{vi}-ETR with the restriction that only polynomially many variables can be used.

Later, in Chapter 8, we will employ a family of satisfiability problems for probabilistic reasoning, which nicely demonstrates the expressiveness of the ETR variants under consideration and illustrates the natural necessity of introducing the summation operator.

Thus, succ-$\exists\mathbb{R}$-completeness seems to be a meaningful yardstick for measuring computational complexity of decision problems. An interesting task would be to investigate problems involving the reals that have been shown to be in EXPSPACE, but not to be EXPSPACE-complete, which are natural candidates for succ-$\exists\mathbb{R}$-complete problems.

Contributions and Structure of this Chapter Below we highlight the main contributions of this chapter.

- We provide the characterization of succ-ETR in terms of nondeterministic real RAMs of exponential time respectively (Section 6.3). Moreover, for the classes over the reals in the sequence of inclusions (6.1), an upward translation result applies, which implies, e.g., NEXP $\subsetneq$ succ-$\exists\mathbb{R}$ unless NP $= \exists\mathbb{R}$ which is widely disbelieved (Section 6.4).
- PSPACE has natural characterizations in terms of ETR; It coincides both with $\exists\mathbb{R}^{\Pi}$ – the class based on ETR enriched with a standard unary product operator, and with succ-$\exists\mathbb{R}_{\text{poly}}$, defined in terms of the succinct variant of ETR with polynomially many variables (Section 6.5).
- $\exists\mathbb{R}^{\Sigma}$ – defined similar to $\exists\mathbb{R}^{\Pi}$, but with the addition of a unary summation operator instead – is contained in PSPACE $= \exists\mathbb{R}^{\Pi}$. We conjecture that this inclusion is strict, as the class is equivalent to $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$, i.e. an $\mathsf{NP}_{\text{real}}$ machine model with access to a $\mathsf{VNP}_{\mathbb{R}}$ oracle (Section 6.6.1).

6.2 Preliminaries

Complexity Classes Based on the ETR The problem succ-ETR and the corresponding class succ-$\exists\mathbb{R}$ are defined in [ZBL23] as follows: succ-ETR is the set of all Boolean circuits C that encode a true sentence φ as follows: Assume that C computes a function $\{0,1\}^N \to \{0,1\}^M$. Then φ is a tree consisting of 2^N nodes, each node being labeled with a symbol of $\{\vee, \wedge, \neg, +, \cdot, <, \leq, =\}$, a constant 0 or 1, or a variable $x_0, \ldots x_{2^N-1}$. For the node $i \in \{0,1\}^N$, the circuit computes an encoding $C(i)$ of the description of node i, consisting of the label of i, its parent, and its two children. The tree represents a true sentence, if the value at the root node would become true after applying the operator of each node to the value of its children, whereby the value of constants and variables are given in the obvious way. As is the case for $\exists\mathbb{R}$, the class succ-$\exists\mathbb{R}$ is then defined as the set of all languages which are polynomial time many-one reducible to succ-ETR.

Besides succ-ETR, [ZBL23] introduce more complete problems for succ-$\exists\mathbb{R}$ as intermediate problems in the hardness proof. Of particular importance is the problem Σ_{vi}-ETR that we already discussed in the introduction. Formally, the problem is defined as an extension of ETR by adding to the signature an additional summation operator $\sum_{x_j=0}^{1}$ with the following semantics: If an arithmetic term is given by a tree with the top gate $\sum_{x_j=0}^{1}$ and $t(x_1, \ldots, x_n)$ is the term computed at the child of the top gate, then the new

term computes $\sum_{e=0}^{1} t(x_1, \ldots, x_{j-1}, e, x_{j+1}, \ldots, x_n)$, that is, we replace the variable x_j by a summation variable e, which then runs from 0 to 1. By nesting the summation operator, we are able to produce a sum with an exponential number of summands. The main reason why the new summation variables are introduced is due to the fact they can be used to *index* the quantified variables x_i. Similarly as in succ-ETR, sentences of Σ_{vi}-ETR allow the use of exponentially many variables, however, the formulas are given directly and do not require any succinct encoding.

6.3 NEXP over the Reals

In [EHM24], Erickson, van Der Hoog, and Miltzow extend the definition of word RAMs to real computations. In contrast to the so-called BSS model of real computation [BSS89], the real RAMs of Erickson et al. provide integer and real computations at the same time, allowing for instance indirect memory access to the real registers and other features that are important to implement algorithms over the reals. The input to a real RAM is a pair of vectors, the first one is a vector of real numbers, the second is a vector of integers. Real RAMs have two types of registers, word registers and real registers. The word registers can store integers with w bits, where w is the word size. The total number of registers is 2^w for each of the two types. Real RAMs perform arithmetic operations on the word registers, where words are interpreted as integers between 0 and $2^w - 1$, and bitwise Boolean operations. On the real registers, only arithmetic operations are allowed. Word registers can be used for indirect addressing on both types of registers and the control flow is established by conditional jumps that depend on the result of a comparison of two word registers or of a real register with the constant 0. For further details we refer to the original paper [EHM24].

The real RAMs of [EHM24] characterize the existential theory of the reals. The authors prove that a problem is in $\exists\mathbb{R}$ iff there is a polynomial time real verification algorithm for it. In this way, real RAMs are an "easy to program" mechanism to prove that a problem is contained in $\exists\mathbb{R}$. Beside the input I, which is a sequence of words, the real verification algorithm also gets a certificate consisting of a sequence of real numbers x and a further sequence of words z. I is in the language if there is a pair (x, z) that makes the real verification algorithm accept. I is not in the language if for all pairs (x, z), the real verification rejects.

Instead of using certificates and verifiers, we can also define nondeterministic real RAMs that can guess words and real numbers on the fly. Like for classical Turing machines, it is easy to see that these two definitions are equivalent (when dealing with time bounded computations).

6.2 Definition. Let $t : \mathbb{N} \to \mathbb{N}$ be a function. We define $\mathsf{NTime}_{\text{real}}(t)$ to be the set of all languages $L \subseteq \{0,1\}^\star$, such that there is a constant $c \in \mathbb{N}$ and a nondeterministic real word-RAM M that recognizes L in time t for all word-sizes $w \geq c \cdot \log(t(n)) + c$.

For any set of functions T, we define $\mathsf{NTime}_{\text{real}}(T) = \bigcup_{t \in T} \mathsf{NTime}_{\text{real}}(t)$. We define our two main classes of interest, $\mathsf{NP}_{\text{real}}$ and $\mathsf{NEXP}_{\text{real}}$ as follows:

$$\mathsf{NP}_{\text{real}} = \mathsf{NTime}_{\text{real}}(\text{poly}(n)), \qquad \mathsf{NEXP}_{\text{real}} = \mathsf{NTime}_{\text{real}}(2^{\text{poly}(n)}).$$

Note that the word size needs to be at least logarithmic in the running time, to be able to address a new register in each step.

One of the main results of Erickson et al. (Theorem 2 in their paper) can be rephrased as $\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$. Their techniques can be extended to prove that $\mathsf{succ}\text{-}\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$.

For completeness sake we will first give a brief overview over their construction for $\mathsf{NP}_{\text{real}} \subseteq \exists\mathbb{R}$ and then expand on it afterwards. Note that while they phrase their result in terms of real verification algorithms, it makes no difference whether we have an additional certificate input – consisting of an integer part and a real part – or whether we can nondeterministically guess both integers and real numbers on the fly. We will highlight this difference in the proof overview when it occurs.

6.3 Lemma ([EHM24, Lemma 11])**.** $\mathsf{NP}_{real} \subseteq \exists\mathbb{R}$.

Proof idea. The construction is similar to standard textbook proofs of Cook's Theorem.

Let M be a nondeterministic real word-RAM M, recognizing some language $L \in \mathsf{NP}_{\text{real}}$ in time $t(n)$ and let $c \in \mathbb{N}$ be the constant from the definition of $\mathsf{NTime}_{\text{real}}$. Further, fix some input $I \in \{0,1\}^n$.[4] We will now construct an ETR formula F that is true iff M with word-size $w := \lceil c \log(t(n)) + c \rceil$ accepts input I in time $T := t(n)$.

[4] [EHM24] work with arbitrary integer inputs, however we believe working just with binary inputs is cleaner and avoids the problem of what to do with inputs that don't fit into individual words, especially since the word-size is part of the algorithm and not of the problem.

In the following, anything denoted $[\![\cdot]\!]$ is considered a variable. The following variables store the state of M at each point in time for $0 \leq t \leq T$.

- For each address i, variable $[\![W(i,t)]\!]$ stores the value of word register $W[i]$ at time t.
- For each address i, variable $[\![R(i,t)]\!]$ stores the value of real register $R[i]$ at time t.
- The variable $[\![pc(t)]\!]$ stores the program counter at time t.

There are now multiple components to the formula F:

Ensuring variables $[\![W(i,t)]\!]$ only ever store words. Remember that any word register can only ever store values between 0 and $2^w - 1$. Checking that some variable contains a word can thus be done by decomposing the value into its w bits and checking whether each such bit is 0 or 1. To allow for the decomposition we introduce global variables $[\![2^i]\!]$ for all $i \in \{0, \ldots, w\}$.[5]

[5] While $[\![2^w]\!]$ is not needed for the decomposition, [EHM24] use it to implement word arithmetic later on, which is computed modulo 2^w.

$$\texttt{PowersOf2} := ([\![2^0]\!] = 1) \wedge \bigwedge_{b=1}^{w} ([\![2^b]\!] = [\![2^{b-1}]\!] + [\![2^{b-1}]\!])$$

Using fresh variables for each decomposition, we can now implement the ETR formula $\texttt{IsWord}(X)$ to constrain a real variable to integers between 0 and $2^w - 1$:

$$\texttt{IsWord}(X) := \exists x_0, \dots, x_{w-1} : \left(X = \sum_{b=0}^{w-1} x_b [\![2^b]\!] \right) \wedge \left(\bigwedge_{b=0}^{w-1} (x_b(x_b - 1) = 0) \right)$$

Using this we can ensure every variable corresponding to a word register contains a word at all times.[6]

$$\texttt{WordsAreWords} := \bigwedge_{i=0}^{2^w-1} \bigwedge_{t=0}^{T} \texttt{IsWord}([\![W(i,t)]\!])$$

[6] [EHM24] only require this for $t = 0$, since they don't allow any nondeterministic guesses afterwards. However, due to T being polynomial in n, this doesn't affect the resulting size too much.

Ensuring the input is stored in the word registers at time 0. We directly hardcode the input $I \in \{0,1\}^n$ into our formula:

$$\texttt{FixInput} := \bigwedge_{i=0}^{n-1} ([\![W(i,0)]\!] = I_i)$$

All other variables corresponding to registers at time 0 are left unconstrained to allow for an initial non-deterministic value on each of them.[7]

[7] This is how [EHM24] allowed inputting of certificates into M. However, since we allow for explicit nondeterministic guesses one could also require all other registers to start with value 0.

Executing a single step of M. Let L denote the number of instructions of M, where line 0 will denote the halting state.[8]

$$\texttt{Execute} := \bigwedge_{t=1}^{T} \bigwedge_{\ell=1}^{L} (([\![pc(t)]\!] = \ell) \implies \texttt{Update}(t, \ell))$$

[8] Halting and accepting will be treated as being the same for the sake of this construction. When they want to reject, [EHM24], directly add a contradiction $0 = 1$ into the formula instead of halting.

This is where we this description will be quite brief and we refer to [EHM24] for the full details of how to implement `Update` for each operation of a real word-RAM. In general $\texttt{Update}(t, \ell)$ will include $[\![W(i,t)]\!] = [\![W(i,t-1)]\!]$ and $[\![R(i,t)]\!] = [\![R(i,t-1)]\!]$ for all registers that are not changed by instruction ℓ and they will advance the program counter by one unless they are a control flow instruction.

We will call specific attention to two kinds of instructions: nondeterministic guessing instructions and indirect memory instructions. To implement a nondeterministic guessing instruction we simply leave the value of the corresponding variable unconstrained.

For the indirect memory instructions we will pick the example of $W[W[i]] \leftarrow W[j]$, all other types of indirect access work analogously (note, we can only use word registers to index registers). The important part here is that while we don't have an indirection mechanism in ETR, we can simply enumerate through all registers and check for the correct one:

$$\bigvee_{k=0}^{2^w-1} (([\![W(i,t-1)]\!] = k) \wedge ([\![W(k,t)]\!] = [\![W(j,t-1)]\!]))$$

The final ETR formula F then has the form:

$$\exists \ldots : \texttt{PowersOf2} \wedge \texttt{FixInput} \wedge \texttt{WordsAreWords} \wedge \texttt{Execute} \\ \wedge ([\![pc(0)]\!] = 1) \wedge ([\![pc(T)]\!] = 0)\,.$$

Note that all indexed $\bigvee_i$ and $\bigwedge_i$ are explicitly expanded for this result. □

If, instead of explicitly expanding the formula at the last step, we encode it succinctly, we get the following simulation result:

6.4 Lemma. *Given a non-deterministic real word-RAM M, an input I, and $w, t \in \mathbb{N}$ with $|I| \leq t$, we can compute a succinct circuit C of size* $\mathrm{poly}(|M| \cdot |I| \cdot w) \cdot \mathrm{polylog}(t)$ *in time* $\mathrm{poly}(|M| \cdot |I| \cdot w) \cdot \mathrm{polylog}(t)$ *encoding an existential formula F over the reals such that F is true iff M with word size w accepts I within t steps.*

Proof. This follows by analyzing Lemma 6.3 carefully. In particular, if we leave out the very last step of expanding the concise $\bigvee_i$ and $\bigwedge_i$, we still have a polynomial sized description of F. This description can directly be translated into a circuit C that represents F succinctly as follows: Each node is a long bitstring and encodes the path taken through F: For example the tuple $(1, 50)$ (encoded as a bitstring) denotes the 50-th child of the 1-st child of the root of F. The children/parents and operation of each node can be directly read off from F. Note that while technically a succ-ETR formula can only contain unary and binary operations, we can always replace a k-ary operation by a binary tree of binary operations. □

Here, for any real RAM M, let $|M|$ denote the size of the encoding of M ("Gödel number") with respect to some standard encoding.

We get the following in analogy to the well-known results that the succinct version of SAT is NEXP-complete.

6.5 Lemma. succ-ETR *is* NEXP_{real}*-complete and thus* $\mathsf{NEXP}_{real} = \mathsf{succ}\text{-}\exists\mathbb{R}$.

Proof. Lemma 6.4 proves that succ-ETR is $\mathsf{NEXP}_{\mathrm{real}}$-hard. Containment of succ-ETR in $\mathsf{NEXP}_{\mathrm{real}}$ follows from expanding the succinct ETR instance and using the fact that nondeterministic real word-RAMs can solve ETR in polynomial time. □

For some of their proofs, Van der Zander et al. [ZBL23] require a useful normal form: They assume that formulas in succ-ETR have contain no negations in the Boolean part and only contain comparison operators $<$ and $=$.[9] For non-succinct instances in ETR, this does not make a difference, since we can push down the negations explicitly using De Morgan's law. For succinct instances, it is not immediately clear how to achieve this, since a path in the Boolean part can be exponentially long. We here show that given a circuit C, an instance of succ-ETR with negations in arbitrary places, we can transform

[9] This is the same normal form we have seen previously in the proof of Lemma 2.7. Indeed, [ZBL23] precisely use this normal form to show succ-ETR-completeness of a succinct version of QUAD.

it into a circuit C' encoding an equivalent succ-ETR instance in polynomial time, that encodes an equivalent instance of succ-ETR with all negations pushed to the bottom.

6.6 Lemma. succ-ETR *is equivalent to* succ-ETR *where the Boolean formula part of the circuit does not contain negations and only contains the comparison operators* $<$ *and* $=$*. Additionally the number of variables in the formula is not affected by this transformation.*

Proof. Let C be a circuit succinctly encoding a formula. We show how we can locally change the structure of the formula encoded by C to not contain negations or different comparison operators than $<$ or $=$. Each replacement gadget has a constant size, so formally we use a constant amount of bits to address which node inside the gadget we are addressing and the remaining bits are used to identify the original node. Since the gadgets are of differing sizes some of those nodes may be unused.

The nodes w computing arithmetic values are replaced by 3 identical copies $w_1, \dots w_3$ of w where every usage of a child c of w is replaced by a usage of c_i in w_i. This step is needed to ensure that the resulting circuit encodes formulas as trees. Every node v computing a Boolean value is replaced by two new nodes v^+ and v^- representing the positive and the negated value of v respectively. We define the construction as follows:

Let v be a node computing an atomic Boolean value. We consider three cases:

v **computes** $s = t$**:** We compute v^+ as $s_1 = t_1$ and v^- as $(s_2 < t_2) \vee (t_3 < s_3)$.

v **computes** $s \leq t$**:** We compute v^+ as $(s_1 < t_1) \vee (s_2 = t_2)$ and v^- as $(t_3 < s_3)$.

v **computes** $s < t$**:** We compute v^+ as $s_1 < t_1$ and v^- as $(t_2 < s_2) \vee (s_3 = t_3)$.

Now, let v be a node labeled with a Boolean operator, with children s and possibly t. We distinguish between the different operator labeling of v:

v **computes** $s \vee t$**:** We compute v^+ as $s^+ \vee t^+$ and v^- as $s^- \wedge t^-$ via De Morgan's law.

v **computes** $s \wedge t$**:** We compute v^+ as $s^+ \wedge t^+$ and v^- as $s^- \vee t^-$ via De Morgan's law.

v **computes** $\neg s$**:** We compute v^+ as $s^- \wedge (0 = 0)$ and v^- as $s^+ \wedge (0 = 0)$. Note that both the $0 = 0$ computations are separate fresh formulas in order to ensure that the result is a formula.

In all cases it is easy to inductively check that v^+ has the same Boolean value as v and v^- computes $\neg v$. Furthermore every node only has one parent, making the construction a formula. All these constructions are local and thus can easily be succinctly encoded by using the original circuit C. □

6.4 The Relationships between the Boolean Classes and Classes over the Reals

Now we study the new class $\mathsf{NEXP}_{\mathrm{real}} = \mathsf{succ}\text{-}\exists\mathbb{R}$ from a complexity theoretic point of view.

$$\mathsf{NP} \subseteq \exists\mathbb{R} \subseteq \mathsf{PSPACE}; \qquad \mathsf{NEXP} \subseteq \mathsf{succ}\text{-}\exists\mathbb{R} \subseteq \mathsf{EXPSPACE}. \tag{6.7}$$

The left side of (6.7) is well-known (see Section 2.5.5). The first inclusion on the right side is obvious, since a real RAM simply can ignore the real part. The second inclusion follows from expanding the succinct instance into an explicit formula (which now has exponential size) and simply using the known PSPACE-algorithm.

We prove two translation results, that is, equality of one of the inclusion in the left equation of (6.7) implies the equality of the corresponding inclusion in the right equation of (6.7).

6.8 Theorem. *If* $\exists\mathbb{R} = \mathsf{NP}$*, then* $\mathsf{succ}\text{-}\exists\mathbb{R} = \mathsf{NEXP}$*.*

Proof. We always have $\mathsf{succ}\text{-}\exists\mathbb{R} \supseteq \mathsf{NEXP}$.

Now we show succ-ETR $\in \mathsf{NEXP}$ assuming $\exists\mathbb{R} = \mathsf{NP}$ which implies $\mathsf{succ}\text{-}\exists\mathbb{R} = \mathsf{NEXP}$. $\exists\mathbb{R} = \mathsf{NP}$ implies that there is a polytime NTM N deciding ETR. We construct an NTM N' as follows:

> Given a circuit C representing a succinct representation of an existential formula φ over the reals, reconstruct φ by evaluating C at all possible inputs. Then check if $\varphi \in$ ETR by using N.

Note that φ has size at most exponential in $|C|$ and as such the usage of N incurs at most a running time exponential in $|C|$. Thus N' nondeterministically decides succ-ETR in time exponential in $|C|$ and we have succ-ETR $\in \mathsf{NEXP}$. □

6.9 Theorem. *If* $\exists\mathbb{R} = \mathsf{PSPACE}$*, then* $\mathsf{succ}\text{-}\exists\mathbb{R} = \mathsf{EXPSPACE}$*.*

Proof. We always have $\mathsf{succ}\text{-}\exists\mathbb{R} \subseteq \mathsf{EXPSPACE}$.

Let A be any language in EXPSPACE decidable in space $s(n)$. Then the padded language $A' = \{x\$1^{s(|x|)} \mid x \in A\}$ is in PSPACE, where \$ is a symbol not used in A. Under the assumption $\exists\mathbb{R} = \mathsf{PSPACE}$ there thus is a polytime reduction f from A' to ETR.

We construct a nondeterministic real RAM M as follows:

> On input x, construct the formula $\varphi = f(x\$1^{s(|x|)})$. Then nondeterministically guess an assignment to all the existentially quantified variables and verify if φ is satisfied.

M uses space polynomial in $|x\$1^{s(|x|)}|$ which is exponential in $|x|$ and decides A. Thus we have shown that $A \in \mathsf{NEXP}_{\mathrm{real}}$. Now the claim follows by Lemma 6.5 and the fact that A was arbitrary. □

The remainder of this section is dedicated to proving a nondeterministic time hierarchy theorem for real word RAMs. Using the characterization of $\exists\mathbb{R}$ and succ-$\exists\mathbb{R}$ in terms of real word RAMs, in particular, we get that succ-ETR is strictly more expressive than ETR:

6.10 Corollary. $\exists\mathbb{R} = \mathsf{NP}_{real} \subsetneq \mathsf{NEXP}_{real} = \text{succ-}\exists\mathbb{R}$.

Let $M_1, M_2, \ldots$ be a fixed efficiently computable enumeration of all nondeterministic real word RAMs. Efficient here means that we can in time $O(i)$ construct a list of all instructions of M_i and store each instruction in $O(1)$ cells of word size $O(\log i)$. We can then access the j-th instruction in constant time.

6.11 Lemma. *There is a nondeterministic real word-RAM U_N that given $i, w, t \in \mathbb{N}$ and a binary input $x \in \{0,1\}^\star$ can simulate running M_i on x with word-size w for t steps in time $T_N \in O(i + t + |x|)$ using any word-size $w_N \geq w + O(\log i + \log t)$.*

Proof. We construct U_N by splitting the memory into two sections, a section A containing 2^w uninitialized real cells and word cells each to store the memory of M_i and a section B of size $O(i + \log t)$ to store the instructions of M_i, the index of the next instruction to be simulated and a counter storing the number of steps left to be simulated, as well as any temporary variables needed to simulate the operations. At the start U_N copies x into the first $|x|$ word cells of section A of the memory and initializes section B to contain the instructions of M_i, the index of the first instruction and the number of steps left as t. This initialization takes time $O(|x| + i + \log t)$. Then U_N simulates M_i step by step as long as there are any steps left. It takes time $O(1)$ to find the next instruction to be simulated. Any access to memory that M_i uses is translated to address into section A instead and every operation is modified to behave as if the cell had word-size w. Afterwards the counter for the steps remaining is decremented. All of this is possible in time $O(1)$, so in total the simulation itself takes $O(t)$ steps.

The only time any of the cells need a word-size (potentially) larger than w is to address both sections A and B of the memory, to store the instructions of M_i or to store the step-counter. For all of these a word-size of $w + O(\log i + \log t)$ is sufficient.

There is no need for any special treatment of nondeterminism, since in this model uninitialized cells automatically are treated as nondeterministically initialized cells. □

We also need to simulate nondeterministic real RAMs using a deterministic real RAM. In the Boolean case, we could simply iterate over all proof strings. This does not work in the real case, since we cannot iterate over all real proof strings. We use an algorithm of Renegar instead (see Lemma 2.9), which is a Boolean algorithm to decide whether an instance of ETR is true. The algorithm is stated as a parallel algorithm, since Renegar used it to show that ETR $\in$ PSPACE (using the fact that parallel polynomial time is polynomial space). It can be transformed into a sequential algorithm using standard methods.

6.12 Lemma. *There is a deterministic real word-RAM U_D that given $i, w, t \in \mathbb{N}$ and a binary input $x \in \{0,1\}^\star$ can simulate running M_i on x with word-size w for t steps in time $T_D \in 2^{\text{poly}(|x|,i,2^w,t)}$ using any word-size $w_D \geq \text{poly}(\log|x|, \log i, 2^w, \log t)$.*

Proof. We use Lemma 6.3 to construct an existential formula F over the reals for M_i with input x being simulated with word-size w for t steps of size $\text{poly}(|x|, i, 2^w, t)$. We can then use Renegar's algorithm (see Lemma 2.9) to deterministically check whether F is satisfiable in time $2^{\text{poly}(|x|,i,2^w,t)}$ for any word-size $w_D \geq \text{poly}(\log|x|, \log i, 2^w, \log t)$. Note that Renegar's algorithm does not need any of the real cells whatsoever and only needs the word-size to address its memory. □

We call a function $t : \mathbb{N} \to \mathbb{N}$ time constructible if there is a deterministic real RAM that, on every input of length n, outputs $t(n)$ in time $O(t(n))$ with word size $O(\log t(n))$.

Note that the following nondeterministic time hierarchy for the real word-RAMs requires the use of O-Notation for both classes due to a lack of acceleration theorems for word-RAMs.

6.13 Lemma. *Let $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ be monotone and time-constructible functions with $t_1(n+1) \in o(t_2(n))$ and $t_2(n) \in \Omega(n)$. Then $\mathsf{NTime}_{real}(O(t_1(n))) \subsetneq \mathsf{NTime}_{real}(O(t_2(n)))$.*

Proof. Note that $\mathsf{NTime}_{\text{real}}(O(t_1(n))) \subseteq \mathsf{NTime}_{\text{real}}(O(t_2(n)))$ holds directly due to the monotonicity of t_1.[10] To show non-equality we use a modification of the diagonalization in [Žák83]. Special care has to be taken in dealing with the word-sizes.

Construct a nondeterministic real word-RAM M as follows:

> On input $1^i01^m01^k$ of length $n = 2 + i + m + k$ compute upper bounds $T := T_D$ and $W := w_D$ for Lemma 6.12 with $|x| = 2 + i + m, i = i, w = \log(t_2(2 + i + m))$ and $t = t_2(2 + i + m)$. Should these values be too big to compute in time $t_2(n)$ and with word-size $\log(t_2(n))$, abort the computation and assume we are always in the first case of what follows.
>
> **if** $k < T$ **or** $\log(t_2(n)) < W$**,** then simulate M_i nondeterministically on $1^i01^m01^{k+1}$ for $t_2(n)$ steps and word-size $\log(t_2(n))$ using U_N from Lemma 6.11. Then return the same result as that simulation.
>
> **if** $k \geq T$ **and** $\log(t_2(n)) \geq W$**,** then simulate M_i deterministically on 1^i01^m0 for $t_2(2+i+m)$ steps and word-size $\log(t_2(2+i+m))$ using U_D from Lemma 6.12. Then invert the result of that simulation.

We see that M uses time $O(t_2(n))$ in both cases. In the first case this follows directly from the time-constructability of t_2 and for the second case it follows from the fact that we only run this case if we can guarantee it will finish in time $O(t_2(n))$. Furthermore M uses a word-size of $O(\log(t_2(n)))$ and will produce the same outputs for any higher word-size. For the first case this follows directly from Lemma 6.11 and $(\log(t_2(n)) + O(\log i + \log(t_2(n))) = O(\log(t_2(n)))$ due to $i \in O(n)$ and $t_2(n) \in \Omega(n)$. For the second

[10] This condition is implicit in the original nondeterministic time-hierarchy proof in [Coo72], by virtue of that proof being only for polynomial time bounds. [SFM78] and [Žák83] resolve this by phrasing their results as existence of a language in the set intersection and not as a strict inclusion. [Žák83] requires $t_2(n+1) \in \Theta(t_2(n))$ whenever they phrase their result as a strict inclusion. Otherwise a simple counterexample where $\mathsf{NTime}_{\text{real}}(O(t_1(n))) \subseteq \mathsf{NTime}_{\text{real}}(O(t_2(n)))$ is violated would be

$$t_1(n) = \begin{cases} n^3 \text{ if } n \text{ is odd} \\ n \text{ otherwise} \end{cases}$$

$$t_2(n) = \begin{cases} n^2 \text{ if } n \text{ is odd} \\ n^4 \text{ otherwise} \end{cases}$$

Note however that we still have $\mathsf{NTime}_{\text{real}}(O(t_2(n))) \setminus \mathsf{NTime}_{\text{real}}(O(t_1(n))) \neq \emptyset$.

case it follows from the fact that we only run this case if we can guarantee that U_D uses a word-size of at most $W \leq \log(t_2(n))$ and at most $T \leq k < n \in O(t_2(n))$ steps. Thus the language L decided by M is contained in $\mathsf{NTime}_{\text{real}}(O(t_2(n)))$.

If we assume $\mathsf{NTime}_{\text{real}}(O(t_1(n))) = \mathsf{NTime}_{\text{real}}(O(t_2(n)))$, there is also an equivalent nondeterministic real word-RAM M' running in time $c_1 \cdot t_1(n) + c_1$ for some $c_1 \in \mathbb{N}$, being correct for every word-size $w \geq c_2 \log(c_1 \cdot t_1(n) + c_1) + c_2$ for some $c_2 \in \mathbb{N}$. Let $i \in \mathbb{N}$ be such that $M_i = M'$ and let $m \in \mathbb{N}$ be such that

$$c_1 \cdot t_1(2 + i + m + k + 1) + c_1 \leq t_2(2 + i + m + k) \tag{6.14}$$

$$\text{and} \quad c_2 \log(c_1 \cdot t_1(2 + i + m + k + 1) + c_1) + c_2 \leq \log(t_2(2 + i + m + k)) \tag{6.15}$$

hold for all $k \in \mathbb{N}$. Such an m exists since $t_1(n + 1) \in o(t_2(n))$. (6.14) ensures that the nondeterministic simulation of M_i is run to completion and (6.15) ensures its result is correct by running the simulation with a big enough word-size. Let T and W be defined as in M and let $K \in \mathbb{N}$ be the smallest integer with $K \geq T$ and $\log(t_2(2+i+m+K)) \geq W$ and such that the computations of T and W succeed in time $t_2(n)$ and with word-size $\log(t_2(n))$. Then we get a contradiction via

$$\begin{aligned}
&1^i01^m0 \text{ is accepted by } M_i \text{ with word-size } \log(t_2(2 + i + m)) \\
\Leftrightarrow &1^i01^m01 \text{ is accepted by } M_i \text{ with word-size } \log(t_2(2 + i + m + 1)) \\
\Leftrightarrow &1^i01^m011 \text{ is accepted by } M_i \text{ with word-size } \log(t_2(2 + i + m + 2)) \\
&\vdots \\
\Leftrightarrow &1^i01^m01^K \text{ is accepted by } M_i \text{ with word-size } \log(t_2(2 + i + m + K)) \geq W \\
\Leftrightarrow &1^i01^m0 \text{ is not accepted by } M_i \text{ with word-size } \log(t_2(2 + i + m))
\end{aligned}$$

Thus the inclusion has to be strict. □

Corollary 6.10 then directly follows.

6.5 Succinct ETR of Polynomially Many Variables

The key feature that makes the language Σ_{vi}-ETR defined in [ZBL23] very powerful is the ability to index the quantified variables in the scope of summation. Nesting the summations allows handling an exponential number of variables. Thus, similarly as in succ-ETR, sentences of Σ_{vi}-ETR allow the use of exponentially many variables, however, the formulas are given directly and do not require any succinct encoding. Due to the fact that variable indexing is possible, [ZBL23] show that Σ_{vi}-ETR is polynomial time equivalent to succ-ETR.

Valiant's class VNP [Bür00; Mah14] is also defined in terms of exponential sums (we recall the definition of VNP and related concepts in Section 2.5.4). However, we cannot

index variables as above, therefore, the overall number of variables is always bounded by the length of the defining expression. It is natural to extend ETR with a summation operator, but without variable indexing as was allowed in Σ_{vi}-ETR. In this way, we can have exponential sums, but the number of variables is bounded by the length of the formula. Instead of a summation operator, we can also add a product operator, or both.

6.16 Definition.

1. Σ-ETR is defined as ETR with the addition of a unary summation operator $\sum_{x_i=0}^{1}$.
2. Π-ETR is defined similar to Σ-ETR, but with the addition of a unary product operator $\prod_{x_i=0}^{1}$ instead.
3. $\Sigma\Pi$-ETR is defined similar to Σ-ETR or Π-ETR, but including both unary summation and product operators.

In the three problems above, the number of variables is naturally bounded by the length of the instance, since the problems are not succinct. The example formula $\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}(x_1+x_2)(x_1+(1-x_2))(1-x_1)=0$ explained in the introduction is also in Σ-ETR and $\Sigma\Pi$-ETR, but not in Π-ETR. The formula $\sum_{e_1=0}^{1}\cdots\sum_{e_N=0}^{1}(x_{\langle e_1,\ldots,e_N\rangle})^2=1$ is in neither of these three languages since it uses variable indexing.

We also consider the succinct version of ETR with only a polynomial number of variables.

6.17 Definition. succ-$\mathrm{ETR}_{\mathrm{poly}}$ is defined similar to succ-ETR, but variables are encoded in unary instead of binary, thus limiting the amount of variables to a polynomial amount of variables. (Note that the given input circuit succinctly encodes an ETR formula and not an arbitrary circuit.)

For succ-ETR, it does not matter whether the underlying structure of the given instance is a formula or an arbitrary circuit, since we can transform the circuit into a formula using Tseitin's trick. This, however, requires a number of new variables that is proportional to the size of the circuit, which is exponential.

Like for ETR, we can now define corresponding classes by taking the closure of the problems defined above. It turns out that we get meaningful classes in this way, however, for some unexpected reason. Except for Σ-ETR, all classes coincide with PSPACE, which we will see below. By restricting the number of variables to be polynomial, the complexity of succ-ETR reduces considerably, from being $\mathsf{NEXP}_{\mathrm{real}}$-complete, which contains NEXP, to PSPACE. On the other hand, the problems are most likely more powerful than ETR, assuming that $\exists\mathbb{R}$ is a proper subset of PSPACE, which is believed by at least some researchers.

6.18 Definition. Let succ-$\exists\mathbb{R}_{\mathrm{poly}}$ be the closure of succ-$\mathrm{ETR}_{\mathrm{poly}}$ under polynomial time many one reductions.

It turns out that the product operator is responsible for bringing the complexity of ETR up to PSPACE. It allows to reduce the PSPACE-complete problem QBF to Π-ETR:

6.19 Lemma. QBF $\leq_P$ Π-ETR.

Proof. Let $Q_1 x_1 \, Q_2 x_2 \ldots Q_n x_n \, \varphi(x_1, \ldots, x_n)$ be a quantified Boolean formula with quantifiers $Q_1, \ldots, Q_n \in \{\exists, \forall\}$. We arithmetize φ as $A(\varphi)$ inductively using the following rules:

φ **is a variable** x_i**:** We construct $A(\varphi) = x_i$.

φ **is** $\neg\varphi_1$**:** We construct $A(\varphi)$ as $1 - A(\varphi_1)$.

φ **is** $\varphi_1 \wedge \varphi_2$**:** We construct $A(\varphi)$ as $A(\varphi_1) \cdot A(\varphi_2)$.

φ **is** $\varphi_1 \vee \varphi_2$**:** We construct $A(\varphi)$ as $1 - (1 - A(\varphi_1)) \cdot (1 - A(\varphi_2))$ via De Morgan's law and the previous two cases.

The special treatment of the $\vee$ operator ensures that whenever $x_1, \ldots, x_n \in \{0, 1\}$, then $A(\varphi)$ evaluates to 1 iff $x_1, \ldots, x_n$ satisfy φ and 0 otherwise. We then arithmetize the quantifiers $Q_1, \ldots, Q_n$ in a similar way, but using the unary product operator.

$Q_i = \forall$**:** We construct $A(\forall x_i \, Q_{i+1} x_{i+1} \ldots Q_n x_n \, \varphi)$ as $\prod_{x_i=0}^{1} A(Q_{i+1} x_{i+1} \ldots Q_n x_n \, \varphi)$

$Q_i = \exists$**:** We construct $A(\exists x_i \, Q_{i+1} x_{i+1} \ldots Q_n x_n \, \varphi)$ as
$1 - \prod_{x_i=0}^{1} (1 - A(Q_{i+1} x_{i+1} \ldots Q_n x_n \, \varphi))$, again using De Morgan's law.

The final Π-ETR formula is then just $A(Q_1 x_1 \, Q_2 x_2 \ldots Q_n x_n \, \varphi) = 1$.

The correctness of the construction follows because a formula of the form $\forall\psi(x)$ is true over the Boolean domain iff $\psi(0) \wedge \psi(1)$ is true. The unary product together with arithmetization allows us to write the whole formula down without an exponential blow-up. □

In the rest of this section, we prove that Π-ETR and the remaining problems discussed above are in PSPACE. We first reduce the problems to succ-$\mathrm{ETR}_{\mathrm{poly}}$ starting with obvious relations:

6.20 Lemma. Σ-ETR $\leq_P$ ΣΠ-ETR *and* Π-ETR $\leq_P$ ΣΠ-ETR.

Showing that succ-$\mathrm{ETR}_{\mathrm{poly}}$ is at least as expressive as ΣΠ-ETR is done using the usual succinct encoding of the formula, included here for the sake of completeness.

6.21 Lemma. ΣΠ-ETR $\leq_P$ succ-$\mathrm{ETR}_{\mathrm{poly}}$.

Proof. Let φ be a $\Sigma\Pi$-ETR instance, encoded as a tree T. We construct a circuit C encoding φ by expanding the summation and product operators explicitly, i.e., for a node $v \in T$ on depth ℓ, we construct 2^ℓ copies of v, indexed as $v_{(e_1,\dots,e_\ell)}$ with $e_i \in \{0,1\}$. Only summation and product operators are counted for the depth ℓ, thus the root node and all nodes in an instance without summation or product operators would have depth 0, and an index $v_{()}$.

If v is a leaf, then $v_{(e_1,\dots,e_\ell)}$ retains the labeling of v, unless the labeling is one of the summation variables of its ancestors. In that case it is replaced by the e_i corresponding to the summation variable. Otherwise if v is an inner node with children s (and possibly t) has any labeling other than the unary sum or product, then $v_{(e_1,\dots,e_\ell)}$ has children $s_{(e_1,\dots,e_\ell)}$ (and $t_{(e_1,\dots,e_\ell)}$).

It remains to construct the case when v is a unary sum or product. If v is a sum gate, then we construct $v_{(e_1,\dots,e_\ell)}$ as the sum of $v_{(e_1,\dots,e_\ell,0)}$ and $v_{(e_1,\dots,e_\ell,1)}$. If v is a product gate, then we construct $v_{(e_1,\dots,e_\ell)}$ as the product of $v_{(e_1,\dots,e_\ell,0)}$ and $v_{(e_1,\dots,e_\ell,1)}$.

The node $v_{()}$ where v is the root of T is the root of the constructed succ-ETR$_{\text{poly}}$ instance. All of this can easily be computed by a circuit C taking as input an encoding of v and an encoding of $(e_1,\dots,e_\ell)$. All of these can be encoded with polynomially many bits in the size of T. □

To show that succ-ETR$_{\text{poly}}$ is in PSPACE, we can again use Renegar's algorithm. It shows more than simply ETR $\in$ PSPACE, because it can handle an exponential number of arithmetic terms of exponential size with exponential degree as long as the number of variables is polynomially bounded. For convenience we will state the exact result by Renegar again:

2.9 Lemma (Theorem 1.1 in [Ren92])**.** *Let $p_1,\dots,p_m$ be polynomials in n variables $x_1,\dots,x_n$ and of degree at most $d \geq 2$ where all coefficients are given as integers of bit length L. Additionally let $\circ_1,\dots,\circ_m \in \{>,<,=\}$ be comparison operators, let P be a Boolean function with m inputs and let $Q_1,\dots,Q_n \in \{\exists,\forall\}$ be quantifiers with ω blocks of quantifiers with sizes $n_1,\dots,n_\omega$[11]. Then there is an algorithm deciding whether $Q_1x_1\,Q_2x_2\,\dots\,Q_nx_n\,P(p_1(x_1,\dots,x_n)\circ_1 0,\dots,p_m(x_1,\dots,x_n)\circ_m 0) = 1$ is a true sentence (when all variables are quantified over the real numbers) in time*

$$L\log(L)\log\log(L)\,(md)^{2^{O(\omega)}\prod_i n_i}$$

using $(md)^{O(n)}$ evaluations of P.

Furthermore there is a parallel algorithm deciding the same in time

$$\log(L)\left(2^\omega\left(\prod_i n_i\right)\log(md)\right)^{O(1)} + Time(P,k)$$

[11] A block of quantifiers of size n_i is a sequence of n_i consecutive quantifiers with the same quantification. For example $\exists\exists\forall\forall\exists$ has $\omega = 3$, $n_1 = 2$, $n_2 = 3$ and $n_3 = 1$.

using $L^2(md)^{2^{O(\omega)}\prod_i n_i}$ *processors and requiring* $(md)^{O(\sum_i n_i)}$ *evaluations of* P*. Here* $Time(P,k)$ *is the parallel time needed for the evaluations of* P *when using* $k(md)^{O(\sum_i n_i)}$ *processors (for any* $k \geq 1$*).*

Renegar states his results in terms of parallel algorithms. However, it is well-known that parallel polynomial time equals polynomial space [FW78].

6.22 Lemma. succ-$\mathrm{ETR}_{\mathrm{poly}} \in$ PSPACE.

Proof. We construct a parallel polynomial time algorithm for succ-$\mathrm{ETR}_{\mathrm{poly}}$: Let C be a circuit computing a function $f : \{0,1\}^N \to \{0,1\}^M$ succinctly encoding an ETR-formula φ containing the variables $x_1, \ldots, x_n$. Construct φ explicitly in parallel using 2^N processors by evaluating f at each possible input. During this construction also check that f encodes a valid ETR-formula, i.e., that all nodes agree with their parents and children, are connected to the root (unless marked as disabled) and there is no Boolean node as a child of an arithmetic node. All checks can be done locally in parallel. The only check here that requires some care is the connectivity to the root, but we can use standard skip list techniques to do this in parallel.

W.l.o.g. we may assume that all comparisons in φ are comparisons with 0 and only use the comparison operators $<$ and $=$. If this is not the case, then we can obtain a formula of this form by using Lemma 6.6 and replacing $a \circ b$ by $a - b \circ 0$ for $\circ \in \{<, =\}$.

Consider the forest obtained by ignoring all Boolean nodes. Each root in this forest corresponds to one of the polynomials in φ. Use the tree contraction algorithm described in [KR90] to compute each of the polynomials in parallel. We apply the tree contraction over the semi-ring of multivariate polynomials, given as lists of coefficients, and compute additions and multiplications of elements using the respective parallel algorithms. We finish the algorithm off by using the parallel algorithm from Lemma 2.9. Whenever this algorithm needs to evaluate the Boolean part of the formula, we again use the standard parallel expression evaluation algorithm using $k = 2^{O(N)}$ processors each.

Note that due to the exponential size of φ we have $m, d, L \leq 2^N$. These bounds require φ to be a formula and not an arbitrary circuit. Additionally in succ-$\mathrm{ETR}_{\mathrm{poly}}$ we have $n \in \mathrm{poly}(N)$ and thus the overall algorithm has a parallel running time of $\mathrm{poly}(N)$ using $2^{\mathrm{poly}(N)}$ processors, implying succ-$\mathrm{ETR}_{\mathrm{poly}} \in$ PSPACE. □

Now Lemmas 6.19 to 6.22 together prove:

6.23 Theorem. PSPACE = succ-$\exists\mathbb{R}_{\mathrm{poly}}$ *and all the problems* Π-ETR*,* $\Sigma\Pi$-ETR*, and* succ-$\mathrm{ETR}_{\mathrm{poly}}$ *are* PSPACE*-complete.*

6.6 ETR with the Standard Summation Operator

In the previous section, we have seen that ETR with a unary product operator (Π-ETR) is PSPACE-complete. Moreover, allowing both unary summation and product operators does not lead to an increase in complexity. In this section, we investigate the complexity of Σ-ETR, ETR with only unary summation operators.

6.24 Definition. Let $\exists\mathbb{R}^{\Sigma}$ be the closure of Σ-ETR under polynomial time many one reductions. Moreover, for completeness, let $\exists\mathbb{R}^{\Pi}$ be the closure of Π-ETR under polynomial time many one reductions.

6.6.1 Machine Characterization of $\exists\mathbb{R}^{\Sigma}$

By Theorem 6.23, we have $\exists\mathbb{R}^{\Pi} = \mathsf{PSPACE}$. For $\exists\mathbb{R}^{\Sigma}$, we can conclude: $\mathsf{NP}_{\text{real}} = \exists\mathbb{R} \subseteq \exists\mathbb{R}^{\Sigma} \subseteq \mathsf{PSPACE}$. We conjecture that all inclusions are strict. In this section, we will provide some arguments in favor of this.

As a warmup, we first observe that using summations we can quite easily solve PP-problems. Even more, we have:

6.25 Lemma. $\mathsf{NP}^{\mathsf{PP}} \subseteq \exists\mathbb{R}^{\Sigma}$.

Proof. The canonical $\mathsf{NP}^{\mathsf{PP}}$-complete problem E-MajSat is deciding the satisfiability of a formula

$$\psi : \exists x_1, \ldots, x_n : \#\{(y_1, \ldots, y_n) \in \{0,1\}^n \mid \phi(x,y) = 1\} \geq 2^{n-1},$$

i.e., deciding whether there is an assignment to the x-variables such that the resulting formula is satisfied by at least half of the assignments to the y-variables [LGM98].

Let $X_1 \ldots X_n, Y_1 \ldots Y_n$ be real variables and $\phi^{\mathbb{R}}$ the arithmetization of ϕ. We build an equivalent $\exists\mathbb{R}^{\Sigma}$ instance as follows:

1. $X_i = 0 \vee X_i = 1$, $1 \leq i \leq n$, and
2. $\sum_{Y_1=0}^{1} \ldots \sum_{Y_n=0}^{1} \phi^{\mathbb{R}}(X,Y) \geq 2^{n-1}$.

Then this instance is satisfiable iff ψ is satisfiable because the X_i are existentially chosen and constraint to be Boolean and $\sum_{Y_1=0}^{1} \ldots \sum_{Y_n=0}^{1} \phi^{\mathbb{R}}(X,Y)$ is exactly the number of satisfying assignments to the Y-variables. □

Similarly to $\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$, we can also characterize $\exists\mathbb{R}^{\Sigma}$ using a machine model instead of a closure of a complete problem under polynomial time many one reductions. For this we define a $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ machine to be an $\mathsf{NP}_{\text{real}}$ machine with a $\mathsf{VNP}_{\mathbb{R}}$ oracle, where $\mathsf{VNP}_{\mathbb{R}}$ denotes the families of polynomials in VNP over the reals. Since $\mathsf{VNP}_{\mathbb{R}}$ is a set of polynomials, the oracle allows us to evaluate a family of polynomials, for example the permanent, at any real input[12]. In order to do so, we add the following operation to a real word-RAM:

$$R[i] \leftarrow O(R[W[j]] \dots R[W[k]]).$$

Recall that for a real word-RAM $R[i]$ denotes a real register, while $W[i]$ denotes a word register. The values i, j, k are assumed to be constant indices. This call evaluates the oracle O by choosing the polynomial in the family with $W[k] - W[j] + 1$ variables and evaluating it with the values of $R[W[j]]$ up to $R[W[k]]$. The result is stored in $R[i]$. In case $W[k] > W[j]$ or the oracle family doesn't contain a polynomial with $W[k]-W[j]+1$ variables[13], the machine rejects.

[12] See Section 2.5.4 for the relevant definitions

[13] This can for example happen for the permanent, where the number of variables is always a perfect square.

The rest of the section is now dedicated to proving

6.26 Theorem. Σ-ETR *is complete for* $\mathsf{NP}_{real}^{\mathsf{VNP}_{\mathbb{R}}}$. *Thus,* $\exists\mathbb{R}^{\Sigma} = \mathsf{NP}_{real}^{\mathsf{VNP}_{\mathbb{R}}}$.

In some sense, $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ is the fully real equivalent to $\mathsf{NP}^{\mathsf{PP}}$, yielding a strengthening of Lemma 6.25. We start by showing

6.27 Lemma. Σ-ETR $\in \mathsf{NP}_{real}^{\mathsf{VNP}_{\mathbb{R}}}$. *This also holds if the* NP_{real} *machine is only allowed to call its oracle once.*

Proof. This proof proceeds in two stages:

1. We normalize the Σ-ETR formula such that all polynomials contained in it have the form $\sum_{Y\in\{0,1\}^m} p(X, Y)$ where p does not contain any unary sums, then
2. translate the formula into a real word-RAM with oracle access.

To normalize the formula we introduce variables $Z_i = \frac{1}{2^i}$ (computed by the real RAM) and iteratively move the unary sums outwards by using the following replacements:

$$\sum_{Y\in\{0,1\}^m} p(X,Y) \cdot \sum_{Y'\in\{0,1\}^{m'}} p'(X,Y') \mapsto \sum_{Y\in\{0,1\}^m} \sum_{Y'\in\{0,1\}^{m'}} p(X,Y)\cdot p'(X,Y')$$

$$\sum_{Y\in\{0,1\}^m} p(X,Y) + \sum_{Y'\in\{0,1\}^{m'}} p'(X,Y') \mapsto \sum_{Y\in\{0,1\}^m} \sum_{Y'\in\{0,1\}^{m'}} (p(X,Y)Z_{m'} + p'(X,Y')Z_m)$$

Note that we choose to introduce the Z_i as variables to reduce the final size of the formulas. A real word-RAM can now simply guess the values X and evaluate the formula. Whenever it needs to evaluate some sum $\sum_{Y\in\{0,1\}^m} p(X, Y)$ it queries the oracle[14]. Since p doesn't contain any unary sums anymore, this polynomial is in $\mathsf{VNP}_{\mathbb{R}}$.

[14] technically all queries have to be to polynomials from the same family, but we can query for any of these using queries to the permanent instead.

To reduce the number of oracle queries we add another two steps before step 1, first turning the formula into a collection of polynomials that has a common zero iff the

formula is satisfiable and then turning this collection of polynomials back into a Σ-ETR-formula with one polynomial, thus only requiring a single oracle call. This is achieved in the same way as Lemma 2.7 (a construction by [SŠ17]) with the difference that we cannot use Tseitin's trick within any of the unary sums. Instead we only use that construction on all nodes in the formula with a Boolean result. We obtain a set of polynomials $p_1, \ldots, p_r$, that still contain unary sums, that have a common zero iff the original formula was satisfiable.

Proceeding with another trick from [SŠ17], $p_1^2 + \ldots + p_r^2 = 0$ is satisfiable iff $p_1, \ldots, p_r$ have a common zero. Continuing with steps 1 and 2 from here on out will then result in a single oracle query containing the entire formula. □

Remains to show that Σ-ETR can simulate any $\mathsf{NP}_{\mathrm{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ machine:

6.28 Lemma. Σ-ETR *is hard for* $\mathsf{NP}_{real}^{\mathsf{VNP}_{\mathbb{R}}}$.

Proof. Since the permanent is $\mathsf{VNP}_{\mathbb{R}}$ complete we w.l.o.g. restrict ourselves to $\mathsf{NP}_{\mathrm{real}}^{\mathrm{per}}$.

We show how to expand on Lemma 6.3 (the construction by [EHM24] showing that ETR is $\mathsf{NP}_{\mathrm{real}}$-hard). In particular, we show how to implement the main part of $\texttt{Update}(t, \ell)$ for an oracle query instruction $R[i] \leftarrow O(R[W[j]] \ldots R[W[k]])$:

$$\bigvee_{a,b} ((\llbracket W(j, t-1) \rrbracket = a) \wedge (\llbracket W(k, t-1) \rrbracket = b) \wedge \llbracket R(i, t) \rrbracket = \texttt{Perm}(t, a, b))$$

where the big or is going over all $a, b \in \{0, \ldots, 2^w - 1\}$ such that $a \leq b$ and $b - a + 1$ is a square number, and $\texttt{Perm}(t, a, b)$ denotes an expression that evaluates the permanent on $\llbracket R(a, t-1) \rrbracket, \ldots, \llbracket R(b, t-1) \rrbracket$.

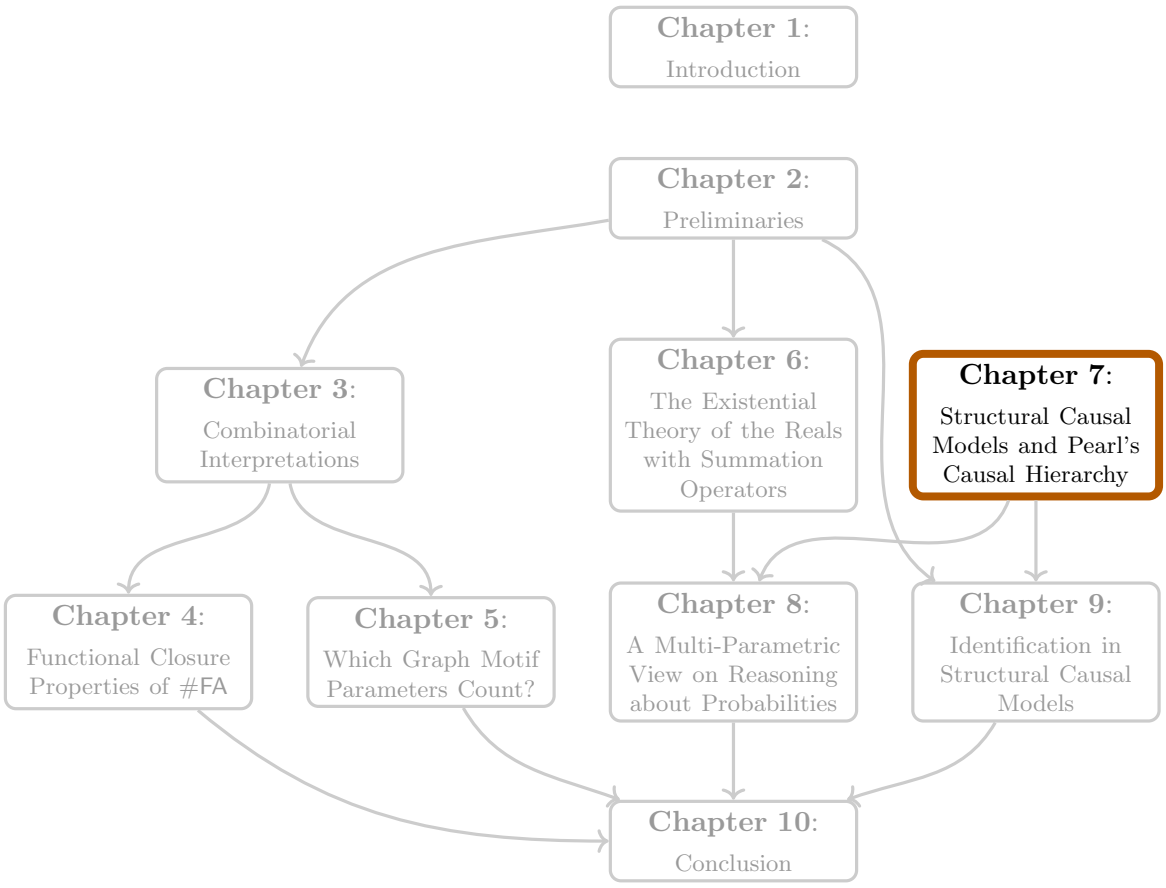
Let $m = \sqrt{b - a + 1}$, then we can evaluate the permanent using

$$\texttt{Perm}(t, a, b) := \sum_{M \in \{0,1\}^{m \times m}} \delta(M) \cdot \prod_{\mu=1}^{m} \sum_{\nu=1}^{m} M_{\mu\nu} \cdot \llbracket R(a + \mu \cdot m + \nu, t-1) \rrbracket$$

where $\delta(M)$ is a polynomial that is 1 iff M is a permutation matrix and 0 otherwise. Given a polynomial $p(x)$ that is 1 if $x = 1$ and 0 for $x \in \{0, \ldots, m\} \setminus \{1\}$, we can use the characterization that a Boolean matrix is a permutation matrix iff each row and each column contains exactly one 1:

$$\delta(M) = \prod_{\mu=1}^{m} p\left(\sum_{\nu=1}^{m} M_{\mu\nu}\right) \cdot \prod_{\nu=1}^{m} p\left(\sum_{\mu=1}^{m} M_{\mu\nu}\right)$$

□



Structural Causal Models and Pearl's Causal Hierarchy

7.1 Pearl's Causal Hierarchy

The development of the modern causal theory in AI and empirical sciences has greatly benefited from an influential structured approach to inference about causal phenomena, which is based on a reasoning hierarchy named "Ladder of Causation", also often referred to as the "Pearl's Causal Hierarchy" (PCH) ([SP08; Pea09; BCII22], see also [PM18] for a gentle introduction to the topic). This three-level framework formalizes various types of reasoning that reflect the progressive sophistication of human thought regarding causation. It arises from a collection of causal mechanisms that model the "ground truth" of *unobserved* nature formalized within a Structural Causal Model (SCM). These mechanisms are then combined with three patterns of reasoning concerning *observed* phenomena expressed at the corresponding layers of the hierarchy, known as *probabilistic* (also called *associational* in the AI literature), *interventional*, and *counterfactual*. This chapter's main goal is to give an informal overview of the concepts and languages used to describe these phenomena, see Section 8.2 for a more formal definition. For simplicity we will restrict our random variables to discrete, finite domains for this exposition.

A basic term at the probabilistic/associational layer is expressed as a common probability, such as $\mathbb{P}(X{=}x \wedge Y{=}y)$ (often abbreviated as $\mathbb{P}(Y{=}y, X{=}x)$ or even just $\mathbb{P}(x, y)$ if the corresponding random variables are clear from the context). This may represent queries like "How likely does a patient have both diabetes ($X{=}x$) and high blood pressure ($Y{=}y$)?" Here we use the usual notation with capital letters and lowercase letters, where the uppercase letters are variables and lowercase letters are an assignment of values to those variables.

The interventional patterns extend the basic probability terms by allowing the use of Pearl's do-operator [Pea09] which models an experiment like a Randomized Controlled Trial [Fis36]. For instance, $\mathbb{P}([x]y)$ which[1], allows to ask hypothetical questions such as, e.g., "How likely is it that a patient's headache will be cured ($Y{=}y$) if they take aspirin ($X{=}x$)?". An example formula at this layer is $\mathbb{P}([x]y) = \sum_z \mathbb{P}(y \mid x, z)\mathbb{P}(z)$ which estimates the causal effect of the intervention $do(X{=}x)$ (all patients take aspirin) on the outcome variable $Y{=}y$ (headache cure). This formula, the prominent back-door adjustment, allows to estimate a causal effect by computing purely observational probabilities and can be used whenever the so-called back-door criterion is fulfilled by the variable Z [Pea09]. Note that an intervention $\mathbb{P}([x]y)$, in general, is a very different concept from the conditional probability $\mathbb{P}(y \mid x)$: The conditional probability only looks at the cases where $X{=}x$ occurred "naturally", while the intervention "forces" $X{=}x$ and observes the causal changes of this change. Coming back to the interventional question "How likely is it that a patient's headache will be cured if they take aspirin?" versus the conditional question "How likely is it that a patient's headache will be cured given that they take aspirin?", in the latter we only aggregate over all cases where patients have taken aspirin, but in the former instead we enforce it for everyone and then look at how this influences the patients' headaches. This distinction will become much more clear once we introduce the underlying structural causal models in Section 7.2.

[1] A common and popular notation for the post-interventional probability is $\mathbb{P}(Y{=}y \mid do(X{=}x))$. In this work, however, we use the notation $\mathbb{P}([X{=}x]Y{=}y)$ since it is more convenient for the analysis of counterfactuals.

The basic terms at the highest level of the hierarchy enable us to formulate queries related to counterfactual situations. For example, $\mathbb{P}([X{=}x]Y{=}y \mid (X{=}x', Y{=}y'))$ expresses the probability that, for instance, a patient who did not receive a vaccine ($X{=}x'$) and died ($Y{=}y'$) would have lived ($Y{=}y$) if he or she had been vaccinated ($X{=}x$). In particular, at this level we are able to not just observe the effects of a single intervention, but instead can compare the outcomes of different interventions.

7.2 Structural Causal Models

A classical probability space consists of a sampling space (the set of all possible atomic events), an event space (all combined events we want to be able to consider), and a probability function (a measure assigning each event a probability between 0 and 1). In case of discrete and finite random variables this simplifies to just an exponential sized table, representing the joint probability distribution over the random variables.

While this definition gives us models that can describe correlations between random variables it is unable to describe any causal effects. To model causal effects we need functional dependencies between the random variables. For example, the random variable X denoting whether a person has lung cancer would depend on the random variable Y denoting whether that same person is smoking. Of course this isn't a deterministic process, so the process determining the value of X would also have its own source of randomness available, additionally to the value of Y.

A *Structural Causal Model* (SCM) as in [Pea09, Sec. 3.2] is a tuple $(\mathcal{F}, P, \mathbf{U}, \mathbf{X})$. It consists of

- the *exogenous* random variables $\mathbf{U} = (U_1, \ldots, U_m)$ – these are unobservable random variables,
- the *endogenous* random variables $\mathbf{X} = (X_1, \ldots, X_n)$ – these are the observable random variables,
- a probability distribution P over the exogenous variables, and
- deterministic functional dependencies $\mathcal{F} = (F_1, \ldots, F_n)$, expressing each endogenous random variable as a function of a subset of the other random variables — both exogenous and endogenous. Formally, this means that for each $i = 1 \ldots n$ we compute the value x_i of variable X_i as $x_i = F_i(\mathbf{pa}_i)$ with $\mathbf{Pa}_i \subseteq \mathbf{X} \cup \mathbf{U}$. The set $\mathbf{Pa}_i$ is called the *parents* of X_i.

In all cases we will assume our models to be *recursive* (also called *semi-Markovian*), i.e. there is some total order $\prec$ of the endogenous variables, s.t. whenever $X_i \not\prec X_j$, then the value of X_i does not depend on the value of X_j.

As long as the deterministic functional dependencies behave "nicely" they naturally lift the probability distribution P from the exogenous variables to the endogenous variables.

As a consequence we write $P_{\mathfrak{M}}(X_1, \ldots, X_n)$ to denote the lifted observable probability distribution, or just $P(X_1, \ldots, X_n)$ if the SCM in question is clear. We will not be going into details about what "nicely" means precisely, however we only concern ourselves with two families of functional dependencies throughout this work where this lifting is clear:

Discrete dependencies: Whenever our random variables are discrete and finite, we represent the functional dependencies by listing the value for each input explicitly. This setting is the focus in Chapter 8.

Notably, while it might not be immediately clear, [ZTB22] have recently proven that whenever all the endogenous variables are discrete and finite, we can w.l.o.g. also assume the exogenous variables to be discrete and finite, albeit with a much larger domain than the endogenous variables.

Linear dependencies: The second setting, the focus of Chapter 9, has real valued variables, gaussian distributed exogenous variables and linear functional dependencies.

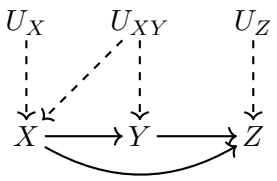
The main benefit of structural causal models is their ability to model causal effects: We can override the values of some endogenous variables – independent of their true values – and then recalculate the remaining endogenous variables using the functional dependencies.

7.2.1 The Graph Structure

While the precise SCM and in particular its functional dependencies are unobservable, a researcher might have some prior knowledge about what random variables influence which other random variables. This notion of influence is captured by a directed graph structure on the random variables, also called the *causal diagram*, induced by the functions F_i: there is an edge from A to B if B depends on A. As long as our model is recursive, this dependency graph is acyclic.

There are a couple variations of this graph structure, depending on the amount of knowledge or assumptions one has about the model:

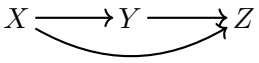
All Random Variables: This is the most versatile model, the graph structure contains both the exogenous and endogenous variables. We highlight dependencies of endogenous variables on exogenous variables as dashed to emphasize that these are influences from unobserved variables. An example structure of an SCM with this representation would be:



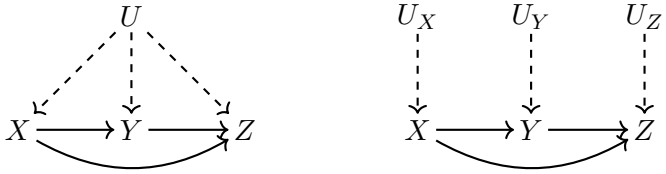
In this representation we generally assume the exogenous variables to be independent and thus all potential[2] (conditional) dependencies can be read off the graph structure using a property called *d-separation* [Pea09].

[2]The dependencies in the graph are purely syntactical dependencies. For example $X = F_X(Z) = Z - Z$ would be a syntactical, albeit degenerate, dependence of X on Z that would still be represented in the graph. The property that all structural (conditional) dependencies (explicit and implicit via d-separation) are true dependencies is called *faithfulness*, which however is not studied in this work.

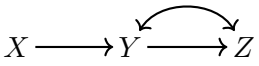
Endogenous Variables Only: We use this representation in two cases: Either we assume that we know nothing about the exogenous variables and in particular their interactions, or – the complete opposite – we assume that each endogenous variable has their own exogenous variable and all exogenous random variables are independent. The latter setting is also called *Markovian*. The structure



can thus imply two different fully specified graphs, depending on the setting:



Endogenous Variables With Correlation: This last representation is used in Chapter 9. Similarly to Markovian models, we assume each endogenous random variable X_i to have its own exogenous random variable U_i. Additionally we have bidirectional edges between X_i and X_j signifying a potential correlation between U_i and U_j. An example is:



Here the exogenous variable U_X influencing X is independent of the exogenous variables U_Y and U_Z influencing Y and Z respectively, while U_Y and U_Z are allowed to be correlated.

7.3 A Full Example

To illustrate the main ideas behind the causality notions, we present in this section a full example that, we hope, will make it easier to understand the formal definitions. In this example, we consider a hypothetical scenario involving three attributes represented by binary random variables: Age, modeled by Z=1 (young) and Z=0 (old), (COVID-19) Vaccination represented by X=1, if yes, and X=0 if not vaccinated, and Recovery, represented by Y=1 (and Y=0 meaning mortality). Below, we describe an SCM that represents an unobserved true mechanism underlying this scenario and illustrates the canonical patterns of reasoning expressible at different levels of the PCH.

U_1	U_2	U_3	$P(\mathbf{u})$
0	0	0	0.0474
0	0	1	0.4266
0	1	0	0.0126
0	1	1	0.1134
1	0	0	0.0316
1	0	1	0.2844
1	1	0	0.0084
1	1	1	0.0756

Z	X	Y
0	1	0
0	1	1
0	0	1
0	0	0
1	0	1
1	0	0
1	1	1
1	1	1

(a) Hidden

Z	X	Y	$P(z,x,y)$
0	0	0	0.1134
0	0	1	0.0126
0	1	0	0.0474
0	1	1	0.4266
1	0	0	0.2844
1	0	1	0.0316
1	1	1	0.0840

(b) Observed P

$P(\mathbf{u})$	Z	X	Y
0.0474	0	1	0
0.4266	0	1	1
0.0126	0	1	0
0.1134	0	1	1
0.0316	1	1	1
0.2844	1	1	1
0.0084	1	1	1
0.0756	1	1	1

(c) $do(X{=}1)$

Z	Y	$P([X=1]z,y)$
0	0	0.06
0	1	0.54
1	0	0.00
1	1	0.40

(d) Post-int. P

Figure 7.1: (a) *Unobserved true nature*: Probabilities of the unobserved variables and the outcomes of the observed variables induced by the mechanism $\mathcal{F}$. (b) *Observational layer*: Probability distribution $P(z,x,y)$ of the observed variables. (c) *Interventional layer*: The modified mechanism $\mathcal{F}$ by using the operator $do(X{=}1)$ (shown in (c)) leads to the *post-interventional* distribution (d), denoted as $P([X{=}1]z,y)$, for $z,y \in \{0,1\}$. A challenging task here is to compute the post-interventional distribution (d) from the observed distribution (b).

The Structural Causal Model: In our example, the model assumes three independent binary random variables $\mathbf{U} = \{U_1, U_2, U_3\}$, with probabilities:

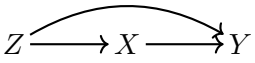
$$P(U_1{=}1) = 0.4$$
$$P(U_2{=}1) = 0.21$$
$$P(U_3{=}1) = 0.9$$

They affect the endogenous (observed) random variables Z, X, Y via the mechanism $\mathcal{F} = \{F_1, F_2, F_3\}$ specified as follows:

$$Z := F_1(U_1) = U_1$$
$$X := F_2(Z, U_2) = ZU_2 + (1-Z)(1-U_2)$$
$$Y := F_3(Z, X, U_3) = XZ + (1-X)(1-U_3) + X(1-Z)U_3$$

Thus, the model determines the distribution $P(\mathbf{u})$, for $\mathbf{u} = (u_1, u_2, u_3)$, and the values for the observed variables, as can be seen in Fig. 7.1(a).

The unobserved random variable U_1 reflects the age of the population and Z is a function of U_1 (which may be more complex in a real population). Getting COVID-19 vaccination depends on age but also on other circumstances (severe allergic reaction to a component of the COVID-19 vaccine, moderate or severe acute illness, etc.) and this is modeled by U_2. So X is a function of Z and U_2. Finally, Y depends on age, being vaccinated, and on further circumstances like having other diseases, which are modeled by U_3. So Y is a function of Z, X, and U_3. Since we assumed the exogenous variables to be independent, this is an example of a Markovian model with the following graph structure:

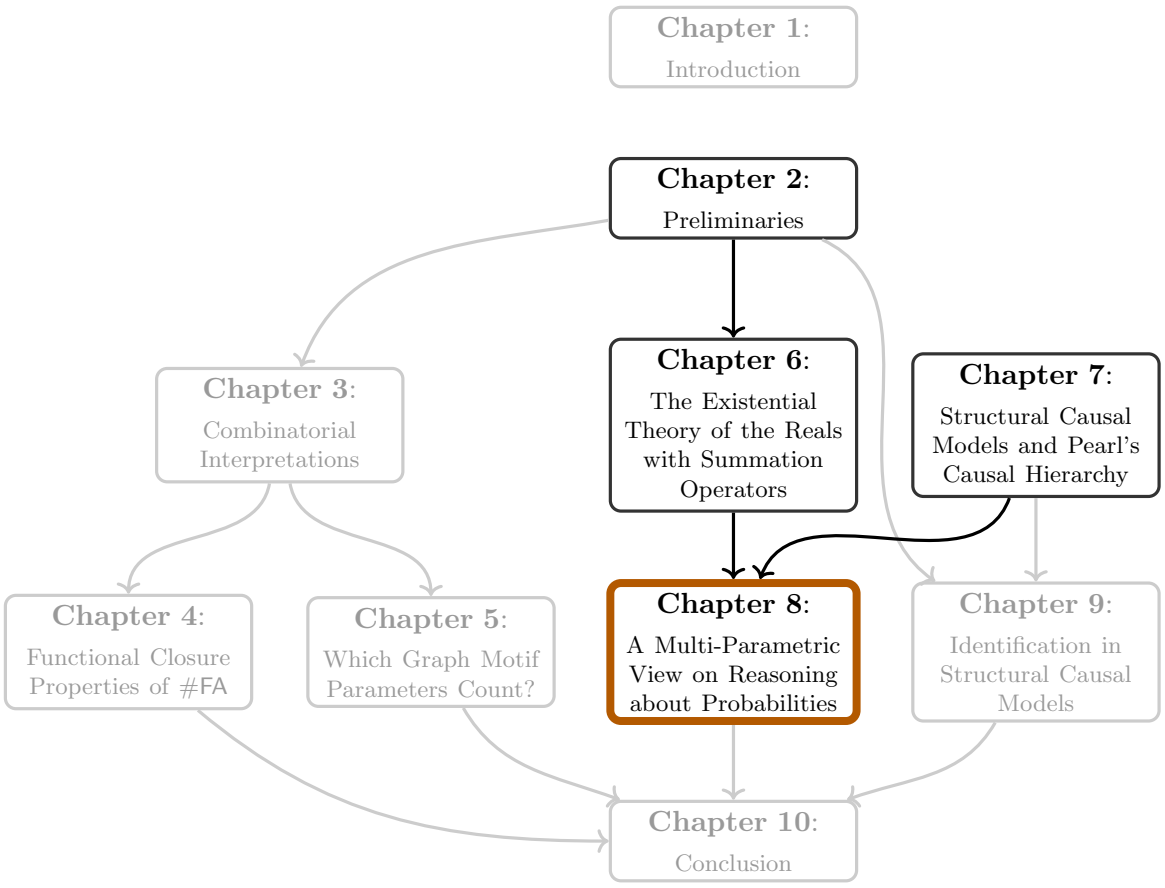


Layer 1 (probabilistic) Empirical sciences primarily rely on observed data, typically represented as probability distributions over measurable variables. In our example, this corresponds to the distribution $P(Z, X, Y)$. The unobserved variables U_1, U_2, U_3 along with the causal mechanism $\mathcal{F}$, remain hidden from direct observation. A researcher thus receives the probabilities (shown in Fig. 7.1(b)) $P(z, x, y) = \sum_{\mathbf{u}} \delta_{\mathcal{F},\mathbf{u}}(z, x, y) \cdot P(\mathbf{u})$, where vectors $\mathbf{u} = (u_1, u_2, u_3) \in \{0, 1\}^3$ and $\delta_{\mathcal{F},\mathbf{u}}(z, x, y) = 1$ if $F_1(u_1){=}z, F_2(z, u_2){=}x$, and $F_3(z, x, u_3){=}y$; otherwise $\delta_{\mathcal{F},\mathbf{u}}(z, x, y) = 0$. The relevant query in our scenario "How likely did vaccinated people recover?", written as the probability $P(Y{=}1 \mid X{=}1)$, can be evaluated as $P(Y{=}1 \mid X{=}1) = P(Y{=}1, X{=}1)/P(X{=}1) = 0.5106/0.558 \approx 0.915$ which says that the probability for recovery ($Y{=}1$) is high given that the patient was vaccinated ($X{=}1$). On the other hand, the query for $X{=}0$ can be evaluated as $P(Y{=}1 \mid X{=}0) = 0.0442/0.442 = 0.1$, which may lead to the opinion that the vaccine is relevant to recovery. However, these results do not reflect a randomized controlled trial [Fis36] that could establish the differences between a COVID-19 vaccination and a placebo vaccination.

Layer 2 (interventional) Consider a randomized trial in which each patient is vaccinated, denoted as $do(X{=}1)$, regardless of age (Z) and other conditions (U_2). We model this by performing a hypothetical intervention in which we replace in $\mathcal{F}$ the mechanism $F_2(Z, U_2)$ by the constant function 1 and leaving the remaining functions unchanged (see, Fig. 7.1(c)). If $\mathcal{F}_{X=1} = \{F_1'{=}F_1, F_2'{=}1, F_3'{=}F_3\}$ denotes the new mechanism, then the *post-interventional* distribution $P([X{=}1]Z, Y)$ is specified as $P([X{=}1]z, y) = \sum_{\mathbf{u}} \delta_{\mathcal{F}_{X=1},\mathbf{u}}(z, y) \cdot P(\mathbf{u})$, where $\delta_{\mathcal{F}_{X=1},\mathbf{u}}$ denotes the function δ as above, but for the new mechanism $\mathcal{F}_{X=1}$ (the distribution is shown in Fig. 7.1(d)).To determine the causal effect of the COVID-19 vaccination on recovery, we compute, in an analogous way, the distribution $P([X{=}0]Z, Y)$ after the intervention $do(X{=}0)$, which means that all patients receive a placebo vaccination. Then, comparing the value $P([X{=}1]Y{=}1) = 0.95$ with $P([X{=}0]Y{=}1) = 0.0874$, we can conclude that $P([X{=}1]Y{=}1) - P([X{=}0]Y{=}1) > 0$. This can be interpreted as a positive (average) effect of the vaccine in the population. Note that it is not obvious how to compute the post-interventional distributions from the observed probability $P(Z, X, Y)$; Indeed, this is a challenging task in the field of causality.

Layer 3 (counterfactual) The key phenomena that can be modeled and analyzed at this level are counterfactual situations. Imagine, e.g., in our scenario, there is a group of patients who have not been vaccinated and died ($X{=}0, Y{=}0$). One may ask, what the outcome (Y) would be had they been vaccinated ($X{=}1$). In particular, one can ask what the probability of recovery would be if we had vaccinated patients in this group. Using the formalism of Layer 3, we can express this as a counterfactual query:

$P([X{=}1]Y{=}1 \mid X{=}0, Y{=}0) = P([X{=}1](Y{=}1) \wedge (X{=}0, Y{=}0))/P(X{=}0, Y{=}0)$. Note that the event $[X{=}1](Y{=}1) \wedge (X{=}0, Y{=}0)$ incorporates simultaneously two counterfactual mechanisms: $\mathcal{F}_{X=1}$ and $\mathcal{F}$. This is the key difference to Layer 2, where we can only have one. Finally, to resolve our question posed at this layer, $P([X{=}1]Y{=}1 \mid X{=}0, Y{=}0) = (0.1134 + 0.2844)/(0.1134 + 0.2844) = 1$, so in our hypothetical scenario every single unvaccinated patient that died would have been saved by the vaccine.



A Multi-Parametric View on Reasoning about Probabilities

8.1 Introduction

Reasoning about probability is essential in many research fields. In computer science, it plays a crucial role in analyzing probabilistic programs, understanding a program's behavior under probabilistic input assumptions, and handling uncertain information in expert systems. In a seminal paper, Fagin et al. [FHM90], introduce a logic to reason about probabilities. Their aim was to formulate a calculus that allows to phrase statements like "$\mathbb{P}(X{=}x) = 1/2$" or "$\mathbb{P}(X{=}x) < \mathbb{P}(Y{=}y)$". They study (among other things) the satisfiability problem for Boolean combinations of such terms, i.e., is there a probability distribution that satisfies all the terms. It turns out that the expressiveness of the underlying calculus has a great influence on the complexity of such satisfiability problems. For instance, in their original work, Fagin et al. investigated three types of terms in the (in)equalities: basic ones, like the one above, which consist of only single basic probabilities, linear combinations of basic probabilities, and polynomial expressions in basic probabilities. This choice parametrizes the satisfiability problem and it turns out that the type of equations has an impact on the complexity of the corresponding satisfiability problem.

We will study the satisfiability of such formulas from a multi-parametric view: there are several parameters in the underlying calculus, like the type of equations mentioned above, and by adjusting these parameters, we get a large family of satisfiability of varying complexities, depending on the choice of our parameters. There has been a lot of work along these lines, see e.g. [FHM90; II20; MII22; IIM24; ZBL23; IIMM24], and the main contribution of this chapter is that we complete the whole picture.

Another dimension is that we can enhance the basic probability terms. This is inspired by the causal theory in AI. The development of the modern causal theory in AI and empirical sciences has greatly benefited from an influential structured approach to inference about causal phenomena, which is based on Pearl's Causal Hierarchy (PCH) we got to know in Chapter 7.

The computational complexity aspects of reasoning about uncertainty in this framework have been the subject of intensive studies in the past decades. The research has resulted in a large number of significant achievements, especially in the case of probabilistic inference with the input probability distributions encoded by Bayesian networks [Pea88]. These problems are of great interest in AI and important results from this perspective include [Coo90; DL93; Rot96; PD04; KF09]. However, despite intensive research, many fundamental problems in the field remain open.

The main interest of our research is focused on the precise characterization of the computational complexity of satisfiability problems for languages of all PCH layers, combined with increasing the expressiveness of (in)equalities by enabling the use of more complex operators. Additionally we investigate the effect of constraining the model in these satisfiability problems, on one hand by fixing the graph structure of the model, and on the other by bounding the model size to be polynomial.

8.1.1 Reasoning about Probabilities: A Multi-Parametric View

One starting point for our investigations is a pioneering paper work by Fagin et al. [FHM90], who introduce a logic to reason about probabilities. We are given a set of random variables over some fixed finite domain D, often $\{0,1\}$. They consider formulas consisting of Boolean combinations of (in)equalities of *basic* and *linear* terms, like $\mathbb{P}((X{=}0 \vee Y{=}1) \wedge (X{=}0 \vee Y{=}0)){=}1 \wedge (\mathbb{P}(X{=}0){=}0 \vee \mathbb{P}(X{=}0){=}1) \wedge (\mathbb{P}(Y{=}0){=}0 \vee \mathbb{P}(Y{=}0){=}1)$ with binary variables X, Y. The authors provide a complete axiomatization for the used logic, which is essentially a formalization of Nilsson's probabilistic logic [Nil86] and they show that the problem of deciding satisfiability is NP-complete. Thus, surprisingly, the complexity is no worse than that of propositional logic. Fagin et al. then extend the language to (in)equalities of *polynomial* terms, with the goal of reasoning about conditional probabilities. They prove that the latter variant is NP-hard and contained in PSPACE. Recently, Mossé et al. [MII22] have given the exact complexity of this problem, showing that deciding the satisfiability is $\exists\mathbb{R}$-complete.

In this chapter, we study the satisfiability problem for such formulas involving probabilities. As already seen above, the choice of the admissible operations have an impact on the complexity. There are further choices, which we can make and which we will discuss in the following. All these choices have an impact on the complexity. Therefore, we will phrase these satisfiability problems as multi-parametric or multi-dimensional problems. The first dimension is the expressiveness of the underlying arithmetic:

- **Arithmetic:** Basic, linear, or polynomial.

The second dimension will be the layer in Pearl's Causal Hierarchy. The setting is of great interest in AI since it allows one not only to reason about *observations*, but also about *causality*, and even *counterfactuals*, which is for instance important in fairness considerations. These three layers form a hierarchy of increasing expressiveness:

- **PCH layer:** Probabilistic (observational), causal (interventional), or counterfactual.

The languages used in [FHM90; MII22] are capable of fully expressing probabilistic reasoning, in particular, they allow to express *marginalization*, which is a common paradigm in this field. However, in the frameworks discussed above, this ability is very limited since the languages *do not* allow the use of a unary summation operator Σ. Thus, for instance[1], to express the marginal distribution of a random variable Y over a subset of (binary) variables $\{Z_1, \ldots, Z_n\}$ as $\sum_{z_1,\ldots,z_n} \mathbb{P}(y, z_1, \ldots, z_n)$, an encoding without summation requires an expansion into $\mathbb{P}(y, Z_1{=}0, \ldots, Z_n{=}0) + \ldots + \mathbb{P}(y, Z_1{=}1, \ldots, Z_n{=}1)$, which is of exponential size in n. Consequently, to analyze the complexity aspects of the studied problems, languages that exploit the standard notation of probability theory and statistics, one requires an encoding that represents marginalization via the sum operator Σ. In the recent paper [ZBL23], the authors present the first systematic study in this setting. They introduce a new natural class, named succ-$\exists\mathbb{R}$, which can be viewed as a succinct variant of $\exists\mathbb{R}$. Alternatively, as we investigated in Chapter 6, succ-$\exists\mathbb{R}$ can

[1] We have chosen this example because it is short. In this particular example, the joint probability could be directly written as $\mathbb{P}(y)$ in the calculus of Fagin et al., see the formal definition in Section 8.2. This is however not true anymore if we sum over more complicated expressions.

be interpreted as the set of problems solvable by non-deterministic real word-RAMs in exponential time, i.e. succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$ in analogy to $\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$ proven by Erickson et al. [EHM24].

Given the impact of the compact marginalization operator, the way we can perform marginalization will be the third dimension:

• **Marginalization:** Expanded or compact.

The fact that the decision problem for basic and linear arithmetic without summation is in NP is not obvious. A model for a formula is a joint probability distribution of n, say, binary variables, which is a table with 2^n entries. Furthermore, it is apriori not clear that the bit size of the entries is polynomial. It turns out, that these problems have what is called a *small model property*. If there is a model, that is, a joint probability distribution satisfying the expression, then there is one which has only polynomially many non-zero entries. Moreover, if the available arithmetic is at most linear we only need a polynomial number of bits to represent each entry. This follows by linear algebra arguments. If the model has both these properties, then it can be guessed efficiently in NP. In the case of polynomial arithmetic, we still have the small model property, but one cannot guarantee that the entries have polynomial size. Furthermore, for polynomial arithmetic, the compact marginalization operator destroys the small model property that was crucial in the work of [FHM90] and [MII22] and in fact increases the complexity of the satisfiability problems dramatically [ZBL23]. So the next dimension is the model size, which can either be polynomially bounded or arbitrary:

• **Model size:** Small (polynomially many entries) or unbounded.

Causal models are often represented as a graph or Bayesian network where all random variables are represented as nodes and the edges encode which variables influence each other. E.g., at the probabilistic level, the graph would encode which variables are conditionally independent of each other. In the satisfiability problems described so far, the graph was not mentioned, so the task was to decide whether there exists any model with any graph structure that satisfies the formula. In many studies, a researcher can infer information about the graph structure, combining observed and experimental data with existing knowledge. Indeed, learning graphical, causal structures from data has been the subject of a considerable amount of research (see, e.g., [Mei+16; DM17; GZS19; SU22] for recent research). Thus, we will study the satisfiability of formulas with the additional requirement that the graph structure G is also given in the input. The goal is to determine whether the formula is satisfied in a model having the structure G.

In fact, such a constraint validity problem[2] is one of the central problems of causality. E.g., given a graph structure, Pearl's prominent do-calculus [Pea09] is often applied to show the validity of the equivalence between causal and probabilistic expressions. Here we have multiple choices: We can constrain the graph on both the endogenous and the exogenous variables or only the endogenous variables. In case we only give the structure

[2] To each of our satisfiability problems there is the corresponding validity problem of whether a given formula is valid for *all* models (potentially of a given graph structure). Each validity problem is thus complete for the co-complexity class of the corresponding satisfiability problem.

of the endogenous variables there are two natural interpretations, the Markovian interpretation where each each endogenous variable has their own independent exogenous variable, and the semi-Markovian interpretation where they might depend arbitrarily on the exogenous variables (however, all the exogenous variables are still independent of each other). Thus, as a fifth dimension, we study the model structure:

- **Model structure:** Specified by a graph as a part of the input or unconstrained.

Due to the sheer amount of potential combinations of these parameters we will not be giving a summary of our results here, see Section 8.7 for a detailed list of all results.

8.2 Preliminaries

In this section, we describe the underlying formalism of the satisfiability problems that we consider. This will cover the first three parameter dimensions: the underlying arithmetic, with or without compact marginalization, and the PCH level. The other two dimensions then naturally follow since they are obtained by restricting the underlying model.

We always consider discrete distributions in the probabilistic and causal languages studied in this paper. We represent the values of the random variables as $\mathit{Val} = \{0, 1, \dots, c-1\}$ and denote by $\mathbf{X}$ the set of random variables used in a system. Here, we use bold letters for sets of variables or sets of values. By capital letters $X_1, X_2, \dots$, we denote the individual variables and assume, w.l.o.g., that they all share the same domain Val.[3] A value of X_i is often denoted by the corresponding lowercase letter x_i or a natural number. In this section, we describe the syntax and semantics of the languages starting with probabilistic ones and then we provide extensions to the causal systems.

[3] This is no restriction. We can always add constraints saying that the probability of the extra values is zero. This becomes more complicated in the higher levels of the PCH, however we can add constraints forcing all occurring interventions to never produce any of the extra values.

By an *atomic* event, we mean an event of the form $X{=}x$, where X is a random variable and x is a value in the domain of X. The language $\mathcal{E}_{\mathit{prop}}$ of propositional formulas over atomic events is defined as the closure of such events under the Boolean operators $\wedge$ and $\neg$. To specify the syntax of interventional and counterfactual events, we define the intervention and extend the syntax of $\mathcal{E}_{\mathit{prop}}$ to $\mathcal{E}_{\mathit{post\text{-}int}}$ and $\mathcal{E}_{\mathit{counterfact}}$, respectively, using the following grammars:

$$\begin{aligned}
\mathcal{E}_{\mathit{prop}} \text{ is defined by } \quad \mathbf{p} &::= X{=}x \mid \neg\mathbf{p} \mid \mathbf{p} \wedge \mathbf{p} \\
\mathcal{E}_{\mathit{int}} \text{ is defined by } \quad \mathbf{i} &::= \top \mid X{=}x \mid \mathbf{i} \wedge \mathbf{i} \\
\mathcal{E}_{\mathit{post\text{-}int}} \text{ is defined by } \quad \mathbf{p_i} &::= [\,\mathbf{i}\,]\,\mathbf{p} \\
\mathcal{E}_{\mathit{counterfact}} \text{ is defined by } \quad \mathbf{c} &::= \mathbf{p_i} \mid \neg\mathbf{c} \mid \mathbf{c} \wedge \mathbf{c}.
\end{aligned}$$

Note that since $\top$ means that no intervention has been applied, we can assume that $\mathcal{E}_{\mathit{prop}} \subseteq \mathcal{E}_{\mathit{post\text{-}int}}$.

The PCH consists of three languages $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, each of which is based on terms of the form $\mathbb{P}(\delta)$. For the (observational or associational) language $\mathcal{L}_1$, we have $\delta \in \mathcal{E}_{\mathit{prop}}$, for the

(interventional) language $\mathcal{L}_2$, we have $\delta \in \mathcal{E}_{post\text{-}int}$ and for the (counterfactual) language $\mathcal{L}_3$, $\delta \in \mathcal{E}_{counterfact}$. The expressive power and computational complexity properties of the languages depend largely on the operations that we are allowed to apply to the basic terms. Allowing gradually more complex operators, we describe the languages which are the subject of our studies below. We start with the description of the languages $\mathcal{T}_i^*$ of terms, with $i = 1, 2, 3$, using the following grammars[4]

[4] In the given grammars, we omit the brackets for readability, but we assume that they can be used in a standard way.

$\mathcal{T}_i^{base}$	$\mathbf{t} ::= \mathbb{P}(\delta_i)$	$\mathcal{T}_i^{base\langle\Sigma\rangle}$	$\mathbf{t} ::= \mathbb{P}(\delta_i) \mid \sum_x \mathbf{t}$
$\mathcal{T}_i^{lin}$	$\mathbf{t} ::= \mathbb{P}(\delta_i) \mid \mathbf{t} + \mathbf{t}$	$\mathcal{T}_i^{lin\langle\Sigma\rangle}$	$\mathbf{t} ::= \mathbb{P}(\delta_i) \mid \mathbf{t} + \mathbf{t} \mid \sum_x \mathbf{t}$
$\mathcal{T}_i^{poly}$	$\mathbf{t} ::= \mathbb{P}(\delta_i) \mid \mathbf{t} + \mathbf{t} \mid -\mathbf{t} \mid \mathbf{t} \cdot \mathbf{t}$	$\mathcal{T}_i^{poly\langle\Sigma\rangle}$	$\mathbf{t} ::= \mathbb{P}(\delta_i) \mid \mathbf{t} + \mathbf{t} \mid -\mathbf{t} \mid \mathbf{t} \cdot \mathbf{t} \mid \sum_x \mathbf{t}$

where δ_1 are formulas in $\mathcal{E}_{prop}$, $\delta_2 \in \mathcal{E}_{post\text{-}int}$, $\delta_3 \in \mathcal{E}_{counterfact}$.

The probabilities of the form $\mathbb{P}(\delta_i)$ are called *primitives* or *basic terms*. In the summation operator $\sum_x$, we have a dummy variable x which ranges over all values $0, 1, \ldots, c-1$. The summation $\sum_x \mathbf{t}$ is a purely syntactical concept which represents the sum $\mathbf{t}[^0/_x] + \mathbf{t}[^1/_x] + \ldots + \mathbf{t}[^{c-1}/_x]$, whereby with $\mathbf{t}[^v/_x]$, we mean the expression in which all occurrences of x are replaced with value v. For example, for $Val = \{0, 1\}$, the expression $\sum_x \mathbb{P}(Y{=}1, X{=}x)$ semantically represents $\mathbb{P}(Y{=}1, X{=}0) + \mathbb{P}(Y{=}1, X{=}1)$.

We note that the dummy variable x is not a (random) variable in the usual sense and that its scope is defined in the standard way.

In the table above, the terms in $\mathcal{T}_i^{base}$ are just basic probabilities with the events given by the corresponding languages $\mathcal{E}_{prop}$, $\mathcal{E}_{post\text{-}int}$, or $\mathcal{E}_{counterfact}$. Next, we extend terms by being able to compute sums of probabilities and by adding the same term several times, we also allow for weighted sums with weights given in unary. Note that this is enough to state all our hardness results. All matching upper bounds also work when we allow for explicit weights given in binary. In the case of $\mathcal{T}_i^{poly}$, we are allowed to build polynomial terms in the primitives. On the right-hand side of the table, we have the same three kinds of terms, but to each of them, we add a marginalization operator as a building block.

The polynomial calculus $\mathcal{T}_i^{poly}$ was originally introduced by Fagin, Halpern, and Megiddo [FHM90] (for $i = 1$) to be able to express conditional probabilities by clearing denominators. While this works for $\mathcal{T}_i^{poly}$, this does not work in the case of $\mathcal{T}_i^{poly\langle\Sigma\rangle}$, since clearing denominators with exponential sums creates expressions that are too large. But we could introduce basic terms of the form $\mathbb{P}(\delta_i \mid \delta)$ with $\delta \in \mathcal{E}_{prop}$ explicitly. All our hardness proofs work without conditional probabilities but many of our matching upper bounds are still true with explicit conditional probabilities.

Expressions like $\mathbb{P}(X{=}1) + \mathbb{P}(Y{=}2) \cdot \mathbb{P}(Y{=}3)$ are valid terms in $\mathcal{T}_1^{poly}$ and expressions liks $\sum_z \mathbb{P}([X{=}0](Y{=}1, Z{=}z))$ and $\sum_z \mathbb{P}(([X{=}0]Y{=}1), Z{=}z)$ are valid terms in the language $\mathcal{T}_3^{poly\langle\Sigma\rangle}$, for example.

Now, let $Lab = \{\text{base}, \text{base}\langle\Sigma\rangle, \text{lin}, \text{lin}\langle\Sigma\rangle, \text{poly}, \text{poly}\langle\Sigma\rangle\}$ denote the labels of all variants of languages. Then for each $* \in Lab$ and $i = 1, 2, 3$, we define the languages $\mathcal{L}_i^*$ of Boolean combinations of inequalities in a standard way:

$$\mathcal{L}_i^* \text{ is defined by } \quad \mathbf{f} ::= \mathbf{t} \leq \mathbf{t}' \mid \neg\mathbf{f} \mid \mathbf{f} \wedge \mathbf{f}, \quad \text{where } \mathbf{t}, \mathbf{t}' \text{ are terms in } \mathcal{T}_i^*.$$

Although the language and its operations can appear rather restricted, all the usual elements of probabilistic and causal formulas can be encoded. Namely, equality is encoded as greater-or-equal in both directions, e.g. $\mathbb{P}(x) = \mathbb{P}(y)$ means $\mathbb{P}(x) \geq \mathbb{P}(y) \wedge \mathbb{P}(y) \geq \mathbb{P}(x)$.

The number 0 can be encoded as an inconsistent probability, i.e., $\mathbb{P}(X{=}1 \wedge X{=}2)$. In a language allowing addition and multiplication, any positive integer can be easily encoded from the fact $\mathbb{P}(\top) \equiv 1$, e.g. $4 \equiv (1+1)(1+1) \equiv (\mathbb{P}(\top) + \mathbb{P}(\top))(\mathbb{P}(\top) + \mathbb{P}(\top))$ while only barely changing the size of the expressions. In languages allowing marginalization we can use a similar trick to generate any powers of c, e.g. $c^4 = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \mathbb{P}(\top)$. Using addition we can then encode any positive integer again.

To define the semantics of the languages, we use structural causal models (see Section 7.2 for the basic notions). Throughout this chapter, an SCM is always a tuple $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$, such that $\mathbf{V} = \mathbf{U} \cup \mathbf{X}$ is a set of variables partitioned into exogenous (unobserved) variables $\mathbf{U} = \{U_1, \ldots, U_m\}$ and endogenous variables $\mathbf{X} = \{X_1, \ldots, X_n\}$.

The tuple $\mathcal{F} = \{F_1, \ldots, F_n\}$ consists of arbitrary functions – often represented as a table of outputs – such that function F_i calculates the value of variable X_i from the values $(\mathbf{x}, \mathbf{u})$ of other variables in $\mathbf{V}$ as $F_i(\mathbf{pa}_i)$. Remember that we assume the domains of endogenous variables $\mathbf{X}$ to be $\{0, 1, \ldots, c-1\}$ and thus to be discrete and finite. In this setting, a priori, the exogenous variables $\mathbf{U}$ could take values in any domain, including infinite and continuous ones. A recent paper [ZTB22] shows, however, that any SCM over discrete endogenous variables is equivalent for evaluating post-interventional probabilities to an SCM where all exogenous variables are discrete with finite domains.

As a consequence, throughout this paper, we assume that domains of exogenous variables $\mathbf{U}$ are discrete and finite, too. Note, however, that this does not imply that the exogenous variables have the same domains as the endogenous variables. In fact, their domains can have sizes up to double exponential in n and c.

For any basic $\mathcal{E}_{int}$-formula $X_i{=}x_i$ (which, in our notation, means $do(X_i{=}x_i)$), we denote by $\mathcal{F}_{X_i=x_i}$ the function obtained from $\mathcal{F}$ by replacing F_i with the constant function $F_i(\mathbf{v}) := x_i$. We generalize this definition for all interventions specified by $\alpha \in \mathcal{E}_{int}$ in the natural way and denote as $\mathcal{F}_\alpha$ the resulting functions. For any $\varphi \in \mathcal{E}_{prop}$, we write $\mathcal{F}, \mathbf{u} \models \varphi$ if φ is satisfied for values of $\mathbf{X}$ calculated from the values $\mathbf{u}$. For $\alpha \in \mathcal{E}_{int}$, we write $\mathcal{F}, \mathbf{u} \models [\alpha]\varphi$ if $\mathcal{F}_\alpha, \mathbf{u} \models \varphi$. And for all $\psi, \psi_1, \psi_2 \in \mathcal{E}_{counterfact}$, we write (i) $\mathcal{F}, \mathbf{u} \models \neg\psi$ if $\mathcal{F}, \mathbf{u} \not\models \psi$ and (ii) $\mathcal{F}, \mathbf{u} \models \psi_1 \wedge \psi_2$ if $\mathcal{F}, \mathbf{u} \models \psi_1$ and $\mathcal{F}, \mathbf{u} \models \psi_2$. Finally, for $\psi \in \mathcal{E}_{counterfact}$, let $S_{\mathfrak{M}}(\psi) = \{\mathbf{u} \mid \mathcal{F}, \mathbf{u} \models \psi\}$.

We define $[\![\mathbf{e}]\!]_{\mathfrak{M}}$, for some expression $\mathbf{e}$, recursively in a natural way, starting with basic terms as follows $[\![\mathbb{P}(\psi)]\!]_{\mathfrak{M}} = \sum_{\mathbf{u}\in S_{\mathfrak{M}}(\psi)} P(\mathbf{u})$ and, for $\delta \in \mathcal{E}_{prop}$, the conditional probability $[\![\mathbb{P}(\psi \mid \delta)]\!]_{\mathfrak{M}} = [\![\mathbb{P}(\psi \wedge \delta)]\!]_{\mathfrak{M}} / [\![\mathbb{P}(\delta)]\!]_{\mathfrak{M}}$, assuming that the expression is undefined if $[\![\mathbb{P}(\delta)]\!]_{\mathfrak{M}} = 0$.[5] For two expressions $\mathbf{e}_1$ and $\mathbf{e}_2$, we define $\mathfrak{M} \models \mathbf{e}_1 \leq \mathbf{e}_2$, if and only if, $[\![\mathbf{e}_1]\!]_{\mathfrak{M}} \leq [\![\mathbf{e}_2]\!]_{\mathfrak{M}}$. The semantics for addition, multiplication, negation and conjunction are defined in the usual way, giving the semantics for $\mathfrak{M} \models \varphi$ for any formula φ in $\mathcal{L}_3^*$.

[5] Remember that conditional probabilities are not formally part of our languages. However, since some of our upper bounds also allow the inclusion of conditional probabilities we add their semantic here for completeness.

8.2.1 Probabilistic and Causal Satisfiability Problems

The (decision) satisfiability problems for languages of the PCH, denoted by $\mathrm{SAT}^*_{\mathcal{L}_i}$, with $i = 1, 2, 3$ and $* \in \mathit{Lab}$, take as input a formula φ in $\mathcal{L}_i^*$ and ask whether there exists a model $\mathfrak{M}$ such that $\mathfrak{M} \models \varphi$.

From the definitions, it is obvious that variants of the problems for any level i are at least as hard as their counterparts at lower levels.

In many studies, a researcher can infer information about the graph structure of the SCM, combining observed and experimental data with existing knowledge. Indeed, learning causal structures from data has been the subject of a considerable amount of research (see, e.g., [DM17; GZS19; SU22] for recent reviews and [RSW21a; NZZZ21; LZY21; Rol+22; Rei+23] for current achievements). Thus, the problem of interest in this setting, which is a subject of this section, is to determine the satisfiability of formulas in SCMs with the additional requirement on the graph structure of the models, also called the *causal diagram*.

The satisfiability problems with an additional requirement on the causal diagram, take as input a formula φ and a DAG $\mathcal{G}$ with the task of deciding whether there exists an SCM with the structure $\mathcal{G}$ for φ. In this way, for problems $\mathrm{SAT}^*_{\mathcal{L}_i}$ with $* \in \mathit{Lab}$, we get the corresponding problems denoted as $\mathrm{SAT}^*_{\mathrm{DAG},\mathcal{L}_i}$, $\mathrm{SAT}^*_{\textsc{Markov},\mathcal{L}_i}$ or $\mathrm{SAT}^*_{\textsc{sMarkov},\mathcal{L}_i}$, respectively depending on whether the causal diagram incorporates all variables or just the endogenous variables (either interpreted as Markovian or as semi-Markovian, see Section 7.2.1).

An important feature of $\mathrm{SAT}^{poly}_{\mathcal{L}_i}$ instances, over all levels $i = 1, 2, 3$, is the small model property which says that, for every satisfiable formula, there is a model whose size is bounded polynomially with respect to the length of the input. This property was used to prove the memberships of $\mathrm{SAT}^{poly}_{\mathcal{L}_i}$ in NP and in $\exists\mathbb{R}$ [FHM90; II20; MII22].

On the other hand, $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathcal{L}_i}$ never allows for a small model, regardless of level. The formula $\sum_{x_1,\ldots,x_n}(\frac{1}{c^n} - \mathbb{P}(x_1,\ldots,x_n))^2 = 0$ is only satisfiable by the uniform distribution on the n variables and thus forces a large model. This motivates investigating what happens if we require a small model regardless. We achieve this by extending an instance with an additional unary input $p \in \mathbb{N}$ and requiring that the satisfying distribution has a support of size at most p. Formally, we use the following:

8.1 Definition (Small Models)**.** The decision problems $\mathrm{SAT}^{*}_{sm,\mathcal{L}_i}$, with $i = 1, 2, 3$ and $* \in \mathit{Lab}$, take as input a formula $\varphi \in \mathcal{L}^*_i$ and a unary encoded number $p \in \mathbb{N}$ and ask whether there exists a model $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U} = \{U_1, \ldots, U_m\}, \mathbf{X})$ such that $\mathfrak{M} \models \varphi$ and $\#\{(u_1, \ldots, u_m) : P(U_1 = u_1, \ldots, U_m = u_m) > 0\} \leq p$.

Combinations of small models and constraints on the causal graph are then defined in the natural way.

Note the subtle difference here to the small model properties that we will see throughout many algorithms: They only show there is *some* model size polynomially bounded with respect to the input length, while our $\mathrm{SAT}^{*}_{sm,\mathcal{L}_i}$ variations require an algorithm to be able to deal with *any specific* model size. However, as it turns out, this subtle difference does not impact the complexity of $\mathrm{SAT}^{*}_{sm,\mathcal{L}_i}$ in any variation we study, although it sometimes requires a separate proof.

8.3 The Complexity of Satisfiability in the PCH: The Base Case

The expressive power of the languages $\mathcal{L}^*_i$, with $* \in \{\mathit{base}, \mathit{lin}, \mathit{poly}\}$ and the layers $i = 1, 2, 3$ has been the subject of intensive research. It is well known, see e.g. [Pea09; BCII22; MII22; SZ81], that they form strict hierarchies along two dimensions: First, on each layer i, the languages $\mathcal{L}^{\mathit{base}}_i, \mathcal{L}^{\mathit{lin}}_i$, and $\mathcal{L}^{\mathit{poly}}_i$ have increasing expressiveness; Second, for every $* \in \{\mathit{base}, \mathit{lin}, \mathit{poly}\}$

$$\mathcal{L}^*_1 \subsetneq \mathcal{L}^*_2 \subsetneq \mathcal{L}^*_3 \tag{8.2}$$

where the proper inclusion means that the language $\mathcal{L}^*_i$ is less expressive than $\mathcal{L}^*_{i+1}$. Note that since adding marginalization does not change the expressiveness of the language $\mathcal{L}^*_i$, the strict inclusions in (8.2) hold also for $* \in \{\mathit{base}\langle\Sigma\rangle, \mathit{lin}\langle\Sigma\rangle, \mathit{poly}\langle\Sigma\rangle\}$.

To prove such a proper inclusion, it suffices to show two SCMs that are indistinguishable in the less expressive language, but that can be distinguished by some formula in the more expressive language. E.g., the SCMs $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ and $\mathfrak{M}' = (\mathcal{F}', P, \mathbf{U}, \mathbf{X})$, with binary variables $\mathbf{U} = \{U_1, U_2\}, \mathbf{X} = \{X_1, X_2\}$, probabilities $P(U_i = 0) = 1/2$, and mechanism $\mathcal{F}$:

$$X_i := U_i,$$

respectively $\mathcal{F}'$:

$$\begin{aligned} X_1 &:= U_1U_2 + (1 - U_1)(1 - U_2) \\ X_2 &:= U_1 + X_1(1 - U_1)U_2, \end{aligned}$$

have the same observable distributions $P_{\mathfrak{M}}(X_1, X_2) = P_{\mathfrak{M}'}(X_1, X_2)$. Thus, $\mathfrak{M}$ and $\mathfrak{M}'$ are indistinguishable in *any* language of the probabilistic layer. On the other hand, after the intervention $X_1 = 1$, we get $P_{\mathfrak{M}}([X_1{=}1]X_2{=}1) = 1/2$ and $P_{\mathfrak{M}'}([X_1{=}1]X_2{=}1) = 3/4$.

Then, e.g., for the $\mathcal{L}_2^{base}$ formula $\varphi : \mathbb{P}([X_1{=}1]X_2{=}1) = \mathbb{P}([X_1{=}1]X_2{=}0)$, we have $\mathfrak{M} \models \varphi$, but $\mathfrak{M}' \not\models \varphi$ which means that φ distinguishes $\mathfrak{M}$ from $\mathfrak{M}'$.

As mentioned in the introduction, a comparison of these languages from the perspective of computational complexity reveals surprisingly different properties than the ones described above. Starting with the base case, i.e. languages without marginalization and no restrictions on the model, it turns out that the complexity depends exclusively on the allowed arithmetic: The satisfiability for the counterfactual level remains the same as for the probabilistic and interventional levels: problems $\text{SAT}_{\mathcal{L}_i}^{base}$ and $\text{SAT}_{\mathcal{L}_i}^{lin}$ for all $i = 1, 2, 3$ are NP-complete, i.e., as hard as reasoning about propositional logic formulas [FHM90; MII22], while $\text{SAT}_{\mathcal{L}_i}^{poly}$ is $\exists\mathbb{R}$-complete [MII22]. Both of these are summarized in Table 8.1.

Terms	Probabilistic	Interventional	Counterfactual
basic	NP (a)		
lin			
poly	$\exists\mathbb{R} = \text{NP}_{\text{real}}$ (b)		

Table 8.1: Completeness results for the base satisfiability problems (a) for $\mathcal{L}_1$ [FHM90] (also see Theorem 8.4), for $\mathcal{L}_2$ and $\mathcal{L}_3$ [MII22] (also see Lemma 8.5), (b) [MII22] (also see Theorem 8.9). The same results hold if we require all models to be small (Theorem 8.13).

We start by including sketched versions of the proofs by [FHM90] and [MII22] here for multiple reasons. First of all: completeness. Our goal is to make this chapter be able to be read in a mostly self-contained manner. Secondly, while they make use of a small model property inside their proofs they don't formally consider the small model as a separate dimension. We will be adding those considerations here. Lastly, they use slightly different models. While [FHM90] doesn't use SCMs at all, [MII22] uses a slightly different version of the calculus with the following differences:

- Their polynomial languages $\mathcal{L}_i^{poly}$ do not allow for subtractions. While at first glance it might seem that simple algebraic modifications could remove subtractions, this is not – at least not obviously – the case for polynomial languages. Consider the expansion of the product $\prod_i(\mathbb{P}(\delta_i) - \mathbb{P}(\delta_i'))$. This expands to a sum of 2^i terms before we are able to move the negative terms to the other side of the (in-)equalities.

- Their endogenous random variables have per-variable domains, while we consider all endogenous domains to be the same set $\{0, \ldots, c-1\}$. This does not effect the expressiveness of the calculus, but it simplifies the usage of the marginalization operator.

- They do not require exogenous variables to be necessarily independent. However, as we will see in Lemma 8.3, this makes no difference until we start restricting the graph structure in Section 8.6.

We adjust the proofs accordingly, also adding in some ideas already that we will be needing later for the consideration of other dimensions.

As a first simplification, we can w.l.o.g. assume all of our SCMs to only have a single exogenous variable.

8.3 Lemma. *Let $\varphi \in \mathcal{L}_i^*$ for $i = 1, 2, 3$ and $\star \in$ Lab. If there is an SCM $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ with $\mathfrak{M} \models \varphi$, then there is also an SCM $\mathfrak{M}' = (\mathcal{F}', P', \mathbf{U}', \mathbf{X})$ with $\mathfrak{M}' \models \varphi$, s.t. $|\mathbf{U}'| = 1$. Moreover, the endogenous parents of each variable in $\mathfrak{M}$ and $\mathfrak{M}'$ are identical.*

Proof. Let $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ be, s.t. $\mathfrak{M} \models \varphi$. We choose $\mathbf{U}' = \{U'\}$ to contain a single random variable with domain being the cartesian product of the domains of the $\mathbf{U}$ with probability distribution $P'((u_1, \ldots, u_m)) := P(u_1, \ldots, u_m)$. Writing $\mathbf{pa}_{i|\mathbf{X}}$ for the vector of values $\mathbf{pa}_i$ restricted to the variables $\mathbf{X}$, the functional dependencies

$$F_i'(\mathbf{pa}_{i|\mathbf{X}}, (u_1, \ldots, u_m)) := F_i(\mathbf{pa}_{i|\mathbf{X}}, \mathbf{pa}_{i|\mathbf{U}})$$

induce the same (counterfactual) distribution for both $\mathfrak{M}'$ and $\mathfrak{M}$ and thus $\mathfrak{M}' \models \varphi$. □

Lemma 8.3 has a second important function: it allows us to ignore the independence of the exogenous variables by virtue of restriction to single exogenous variables.

8.4 Theorem (Theorem 2.9 in [FHM90])**.** $\mathrm{SAT}_{\mathcal{L}_1}^{base}, \mathrm{SAT}_{\mathcal{L}_1}^{lin}$ *are* NP*-complete.*

Proof idea. To show NP-hardness, in a 3-CNF formula φ, replace each positive literal x_i by X_i=1 and each negative literal $\overline{x_i}$ by $X_i = 0$ to obtain φ'. It is now easy to see that $\mathbb{P}(\varphi') > 0$ has a model with binary random variables X_i iff φ is satisfiable.

Showing containment in NP proceeds in two stages. Let $\varphi \in \mathcal{L}_1^{lin}$. First we show that if there is any SCM $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ with $\mathfrak{M} \models \varphi$, then there is also an SCM $\mathfrak{M}' = (\mathcal{F}', P', \mathbf{U}', \mathbf{X})$ with $\mathfrak{M}' \models \varphi$ where $\mathfrak{M}'$ only has a single exogenous variable $\mathbf{U}' = \{U'\}$ with domain size at most polynomial in the size of φ. Moreover, all functional dependencies only depend on U' and all probabilities occurring in P' are rational numbers with binary representation of polynomial size. Overall, this implies that $\mathfrak{M}'$ only takes polynomially many bits to be represented. This allows to simply guess $\mathfrak{M}'$ non-deterministically as a second stage and to then verify $\mathfrak{M}' \models \varphi$.

By Lemma 8.3 we can construct a first version of $\mathfrak{M}'$ with $|\mathbf{U}'| = 1$. Remains to show how to limit the domain of U' and the representation of P' to polynomial size. In φ, replace each basic term $\mathbb{P}(\delta)$ by the sum $\sum_{u' \models \delta} P(u')$. Considering all the different $P(u')$ as variables and adding the constraints $\sum_{u'} P(u') = 1$ and $0 \leq P(u') \leq 1$ for all values u', we get a linear system with a polynomial amount of (in-)equalities. Thus if there is any

solution (which we initially assumed to exist with $\mathfrak{M} \models \varphi$), there is also a solution where only polynomially many values u' have $P(u') > 0$ and all $P(u')$ are rational numbers with polynomially sized binary representations. For the precise bounds on the bit-lengths, see Lemma 4.10 in [FHM90]. □

Mossé et al. [MII22], as a first result, build on top of this result by lifting it across the PCH. Note that while their result uses – a bit unconventionally – non-deterministic polynomial time reductions, both NP and $\exists\mathbb{R}$ are closed under them (see Theorem 4.11 in [TKO13] for this result about $\exists\mathbb{R}$).

8.5 Lemma (Proposition 4.2 in [MII22])**.** $\mathrm{SAT}^*_{\mathcal{L}_3} \leq_{NP} \mathrm{SAT}^*_{\mathcal{L}_1}$ *for* $* \in \{base, lin, poly\}$.

Proof idea. This reduction heavily builds on the concept of complete state descriptions: A complete state description ϵ contains for all possible interventions and variables what value those variables take on with that intervention.

Let $\varphi \in \mathcal{L}^*_3$. We define the set of relevant state descriptions Δ_φ where all interventions are restricted to those actually appearing in φ.[6] Let $\mathcal{I}$ be the set of all interventions appearing in φ. Then

$$\Delta_\varphi := \left\{ \bigwedge_{\alpha\in\mathcal{I}} \left([\alpha] \bigwedge_{i=1}^{n} (X_i{=}x_i^\alpha) \right) \,\middle|\, x_i^\alpha \in \{0,\dots,c-1\} \right\}.$$

[6] This is where we deviate from the original proof a bit. Mossé et al. only consider assignments that actually appear in φ. However the size of each individual relevant state description remains polynomial size, even when considering the full domain for each variable.

Any distinct ϵ_1, ϵ_2 in Δ_φ are disjoint events. Denote the subset of Δ_φ compatible with a variable ordering $\prec$ of the endogenous variables as $\Delta_{\varphi,\prec}$. This compatibility can be checked in polynomial time by Lemma 4.5 in [MII22].

The reduction now proceeds as follows:

1. Guess a variable ordering $\prec$ of the endogenous variables $\mathbf{X}$.
2. Guess a subset of state descriptions $\Delta' \subseteq \Delta_{\varphi,\prec}$ of size $|\varphi|$.
3. Replace each event $\mathbb{P}(\delta)$ in φ by $\sum_{\substack{\epsilon\in\Delta'\\ \epsilon\models\delta}} \mathbb{P}(\epsilon)$.
4. Replace all occurring state descriptions ϵ by disjoint probabilistic events.

To argue the correctness consider an SCM $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ with $\mathfrak{M} \models \varphi$ and the following system of equations in the non-negative unknowns $\mathbb{P}(\epsilon)$ for $\epsilon \in \Delta_{\varphi,\prec}\}$:

$$\sum_{\epsilon\in\Delta_{\varphi,\prec}} \mathbb{P}(\epsilon) = 1 \tag{8.6}$$

$$\sum_{\substack{\varepsilon\in\Delta_{\varphi,\prec}\\ \epsilon\models\delta}} \mathbb{P}(\epsilon) = \mathbb{P}_{\mathfrak{M}}(\delta) \qquad \text{for all } \delta \text{ occurring in } \varphi \tag{8.7}$$

This system has at most $|\varphi|$ equations and thus a solution with $\mathbb{P}(\epsilon) > 0$ for at most $|\varphi|$ many ϵ by Lemma 4.8 in [FHM90]. □

Mossé et al. then finish the proof by showing that $\mathrm{SAT}^{poly}_{\mathcal{L}_1}$ is $\exists\mathbb{R}$ complete. It uses the characterization $\exists\mathbb{R} = \mathsf{NP}_{\mathrm{real}}$ (see Chapter 6 and [EHM24]) to simplify the upper bound.

8.8 Theorem (Theorem 1 in [MII22])**.** $\mathrm{SAT}^{poly}_{\mathcal{L}_1}$ *is* $\exists\mathbb{R}$*-complete.*

Proof idea. We start with $\exists\mathbb{R}$ containment. This is done by giving an $\mathsf{NP}_{\mathrm{real}}$ algorithm deciding $\mathrm{SAT}^{poly}_{\mathcal{L}_1}$. Let $\varphi \in \mathcal{L}_1^{poly}$. By the proof of Lemma 8.5 we can w.l.o.g. assume that φ contains at most $|\varphi|$ many different events $\varepsilon_1, \dots, \varepsilon_k$, all of which are disjoint. A non-deterministic real word RAM can now guess the real values of $P(\varepsilon_i)$, verify their non-negativity and $\sum_i P(\varepsilon_i) = 1$ and check whether they fulfill φ.[7]

[7] Note that while Mossé et al. never considered subtractions, a non-deterministic real word RAM can do subtractions.

To prove the $\exists\mathbb{R}$-hardness of $\mathrm{SAT}^{poly}_{\mathcal{L}_1}$, consider the $\exists\mathbb{R}$-complete problem ETR-INV (see [AAM18] a completeness proof): For variables $x_1, \dots, x_n$ and equations of the form $x_i + x_j = x_k$ or $x_i x_j = 1$, decide whether there are reals $x_1, \dots, x_n \in [1/2, 2]$ satisfying the equations.

The polynomial time reduction from ETR-INV to $\mathrm{SAT}^{poly}_{\mathcal{L}_1}$ constructs the constraints

$$
\begin{aligned}
&\frac{1}{n} \geq \mathbb{P}(\delta_i) = \mathbb{P}(\delta_i') \geq \frac{1}{4n} && \text{for } i = 1, \dots, n \\
&\mathbb{P}(\delta_i \wedge \delta_j) = \mathbb{P}(\delta_i) \cdot \mathbb{P}(\delta_j) && \text{for } i, j = 1, \dots, n \\
&\mathbb{P}(\delta_i \wedge \delta_j) = \frac{1}{4n^2} && \text{for each equation } x_i x_j = 1 \\
&\mathbb{P}(\delta_i' \vee \delta_j') = \mathbb{P}(\delta_k') && \text{for each equation } x_i + x_j = x_k
\end{aligned}
$$

where the δ_i and δ_i' are each fresh events, corresponding to the variable x_i, with the δ_i' being disjoint from each other (and the δ_i being pairwise independent by the second set of equations).

To implement the constants $\frac{1}{N}$, replace each of them by $\mathbb{P}(\epsilon_N^{(N)})$ for some fresh disjoint events $\epsilon_1^{(N)}, \dots, \epsilon_N^{(N)}$ with $\mathbb{P}(\bigwedge_i \epsilon_i^{(N)}) = 1$ and $\mathbb{P}(\epsilon_i^{(N)}) = \mathbb{P}(\epsilon_j^{(N)})$ for $i, j = 1, \dots, N$.

The correspondence $\mathbb{P}(\delta_i) = \mathbb{P}(\delta_i') = \frac{x_i}{2n}$ allows to transfer solutions between the two instances. □

In conclusion, combining Theorems 8.4 and 8.8 with Lemma 8.5 gives us

8.9 Theorem. *For any* $i = 1, 2, 3$, $\mathrm{SAT}^{base}_{\mathcal{L}_i}$ *and* $\mathrm{SAT}^{lin}_{\mathcal{L}_i}$ *are* NP*-complete and* $\mathrm{SAT}^{poly}_{\mathcal{L}_i}$ *is* $\exists\mathbb{R}$*-complete.*

Mossé et al. only consider conditional probabilities as a restricted version of polynomial terms and as a motivation for then considering polynomial terms. As a final strengthened upper bound we show that combining both is also possible and does not change the computational complexity.

8.10 Lemma. $\mathrm{SAT}^{poly}_{\mathcal{L}_3} \in \exists\mathbb{R}$ *is also true if the basic terms are allowed to contain conditional probabilities.*

Proof. We define conditional probabilities $\mathbb{P}(\delta|\delta')$ to be undefined if $\mathbb{P}(\delta') = 0$ (this proof works similarly for other definitions). In the input formula φ, replace $\mathbb{P}(\delta|\delta')$ by $\frac{\mathbb{P}(\delta \wedge \delta')}{\mathbb{P}(\delta')}$.

Again, we use the ideas from Lemma 8.5 and Theorem 8.8 to w.l.o.g. assume that φ contains at most $|\varphi|$ many different events $\varepsilon_1, \ldots, \varepsilon_k$, all of which are disjoint. Additionally to the checks in Theorem 8.8, the non-deterministic real word RAM can now also verify that all of the resulting divisions have non-zero dividends.

Even more, this way we could also add arbitrary divisions to our formulas. □

8.3.1 Small Models

As we have seen, all upper bounds of the previous section crucially depend on the small-model property. In fact, the proof of Lemma 8.5 shows, if there is any model satisfying φ, then there is always also a model satisfying φ with size bounded by $|\varphi|$. This gives rise to a deterministic polynomial time reduction to the small model variant of each problem.

8.11 Corollary. $\mathrm{SAT}^{*}_{\mathcal{L}_i} \leq_P \mathrm{SAT}^{*}_{sm,\mathcal{L}_i}$ *for any* $i = 1, 2, 3$ *and* $* \in \{base, lin, poly\}$.

The proof of Lemma 8.5 then proceeds and converts causal terms to purely probabilistic ones. However, the small model property also allows us to decide $\mathrm{SAT}^{lin}_{\mathcal{L}_3}$ and $\mathrm{SAT}^{poly}_{\mathcal{L}_3}$ more directly. As an added benefit, this algorithm works for arbitrary bounds on the model size.

8.12 Lemma. $\mathrm{SAT}^{lin}_{sm,\mathcal{L}_3}$ *is in* NP *and* $\mathrm{SAT}^{poly}_{sm,\mathcal{L}_3}$ *is in* $\exists\mathbb{R}$.

Proof. Lemma 8.3 does not change the size of the model, so we can w.l.o.g. assume that there is only a single exogenous variable U with domain size bounded by the size of the model p. The algorithm starts by guessing a causal order $X_1 \prec \ldots \prec X_n$ of the endogenous random variables and a probability distribution on U with polynomial bit-length[8] for each probability in the case of $\mathrm{SAT}^{lin}_{sm,\mathcal{L}_3}$ and as real numbers in the case of $\mathrm{SAT}^{poly}_{sm,\mathcal{L}_3}$. It then evaluates the input formula φ by summing over all values of U at each term. Throughout this evaluation it incrementally builds a partial model of the deterministic dependencies, respecting the causal order $\prec$. Whenever it encounters any atomic event under an intervention that cannot be answered by this partial model yet, it non-deterministically guesses the corresponding entries and stores them.

Since φ only contains polynomially many different interventions and atomic events, this partial model remains polynomially sized. □

[8] The precise bound on the bit-length can again be determined using Lemma 4.10 in [FHM90]. While we guess the causal dependencies in the following, the relevant equations remain linear in the probabilities of U.

Finally, Corollary 8.11 together with Theorem 8.9 prove the corresponding hardness for our small model variants and we get:

Terms	Probabilistic	Interventional	Counterfactual
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$	PSPACE	NEXP
lin & marg.			

Table 8.2: Completeness results for the linear satisfiability problems with added marginalization. All these results apply to both, the unconstrained model and the small model variants. See Theorem 8.18 ($\mathcal{L}_1$), Theorem 8.24 ($\mathcal{L}_2$), and Theorem 8.33 ($\mathcal{L}_3$) for the results on unconstrained models and Theorem 8.19 ($\mathcal{L}_1$), Theorem 8.25 ($\mathcal{L}_2$), and Theorem 8.34 ($\mathcal{L}_3$) for the results on small models.

8.13 Theorem. *For any* $i = 1, 2, 3$, $\mathrm{SAT}^{base}_{sm,\mathcal{L}_i}$ *and* $\mathrm{SAT}^{lin}_{sm,\mathcal{L}_i}$ *are* NP*-complete and* $\mathrm{SAT}^{poly}_{sm,\mathcal{L}_i}$ *is* $\exists\mathbb{R}$*-complete.*

8.4 The Impact of Marginalization on Linear Languages

In Section 8.3 we have seen that the level of the PCH does not impact the computational complexity of satisfiability problems without marginalization. In this section, we show that the situation changes drastically once we introduce marginalization: The satisfiability problems $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_i}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_i}$ become $\mathsf{NP}^{\mathsf{PP}}$-, PSPACE-, resp. NEXP-complete depending on the level i (see Table 8.2). This demonstrates the first strictly increasing complexity of reasoning across the PCH, assuming the widely accepted assumption that the inclusions $\mathsf{NP}^{\mathsf{PP}} \subseteq \mathsf{PSPACE} \subseteq \mathsf{NEXP}$ are proper. This section focuses on the basic and linear languages across the PCH, while the case of polynomial languages will be discussed in Section 8.5 separately.

8.4.1 The Probabilistic (Observational) Level

Marginalization via the summation operator, combined with the language expressing events δ on a specific level of the PCH, increases the complexity of reasoning to varying degrees, depending on the level. In the probabilistic case, it jumps from NP to $\mathsf{NP}^{\mathsf{PP}}$-completeness since the atomic terms $\mathbb{P}(\delta)$ can contain Boolean formulas δ, which, combined with the summation operator, allows to count the number of all satisfying assignments of a Boolean formula by summing over all possible values for the random variables in the formula. Determining this count is the canonical #P-complete problem, so evaluating the equations given a model is #P-hard. From this, together with the need of finding a model, we will conclude in this section that $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_1}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_1}$ are $\mathsf{NP}^{\mathsf{PP}}$-complete, which is the same as $\mathsf{NP}^{\#\mathsf{P}}$-completeness.

As a reminder, $\mathsf{NP}^{\mathsf{PP}}$ is the class of problems decidable by polynomial time non-deterministic Turing machines that have access to a PP oracle.

We start with a technical but useful fact that a sum in the probabilistic language can be partitioned into a sum over probabilities and a sum over purely logical terms. This generalizes the property shown by [FHM90] in Lemma 2.3 for languages without the summation operator.

8.14 Fact. Let $\delta \in \mathcal{E}_{prop}$ be a propositional formula over variables $X_{i_1}, \ldots, X_{i_l}$. A sum $\sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \mathbb{P}(\delta)$ is equal to $\sum_{\hat{x}_1} \ldots \sum_{\hat{x}_n} p_{\hat{x}_1 \ldots \hat{x}_n} \sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l})$ where the range of the sums is the entire domain, $p_{\hat{x}_1 \ldots \hat{x}_n}$ is the probability of $\mathbb{P}(X_1{=}\hat{x}_1 \wedge \ldots \wedge X_n{=}\hat{x}_n)$ and $\delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l})$ is a function that returns 1 if the implication $(X_1{=}\hat{x}_1 \wedge \ldots \wedge X_n{=}\hat{x}_n) \rightarrow \delta(x_{i_1}, \ldots, x_{i_l})$ is a tautology and 0 otherwise.

Proof. $\sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \mathbb{P}(\delta)$ is equivalent to

$$
\begin{aligned}
&\sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \sum_{\hat{x}_1} \ldots \sum_{\hat{x}_n} \mathbb{P}((X_1{=}\hat{x}_1 \wedge \ldots \wedge X_n{=}\hat{x}_n) \wedge \delta) \\
=&\sum_{\hat{x}_1} \ldots \sum_{\hat{x}_n} \sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \mathbb{P}((X_1{=}\hat{x}_1 \wedge \ldots \wedge X_n{=}\hat{x}_n) \wedge \delta) \\
=&\sum_{\hat{x}_1} \ldots \sum_{\hat{x}_n} \sum_{x_{i_1}} \ldots \sum_{x_{i_l}} p_{\hat{x}_1 \ldots \hat{x}_n} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l}) \\
=&\sum_{\hat{x}_1} \ldots \sum_{\hat{x}_n} p_{\hat{x}_1 \ldots \hat{x}_n} \sum_{x_{i_1}} \ldots \sum_{x_{i_l}} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l}),
\end{aligned} \tag{8.15}
$$

which completes the proof. □

The canonical satisfiability problem SAT, where instances consist of Boolean formulas in propositional logic, plays a key role in a broad range of research fields, since all problems in NP, including many real-world tasks can be naturally reduced to it. As discussed earlier, $\mathrm{SAT}^{base}_{\mathcal{L}_1}$ and $\mathrm{SAT}^{lin}_{\mathcal{L}_1}$ do not provide greater modeling power than SAT. Enabling the use of summation significantly changes this situation: below we demonstrate the expressiveness of $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_1}$, showing that any problem in $\mathsf{NP}^{\mathsf{PP}}$ can be reduced to it in polynomial time.

8.16 Lemma. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_1}$ *is* $\mathsf{NP}^{\mathsf{PP}}$*-hard.*

Proof. The canonical $\mathsf{NP}^{\mathsf{PP}}$-complete problem E-MajSat is deciding the satisfiability of a formula $\psi : \exists x_1 \ldots x_n : \#y_1 \ldots y_n \phi \geq 2^{n-1}$, i.e. deciding whether the Boolean formula ϕ has a majority of satisfying assignments to Boolean variables y_i after choosing Boolean variables x_i existentially [LGM98]. We will reduce this to $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_1}$. Let there be $2n$ random variables $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ associated with the Boolean variables. Assume w.l.o.g. that these random variables have domain $\{0, 1\}$. Let ϕ' be the formula ϕ

after replacing Boolean variable x_i with $X_i = 0$ and y_i with $Y_i = y_i$. Note that this is not representing an assignment directly, but rather an "xor-mask" modifying what assignment the X_i and Y_j represent.

Consider the probabilistic inequality $\varphi : \sum_{y_1} \dots \sum_{y_n} \mathbb{P}(\phi') \geq 2^{n-1}$ whereby 2^{n-1} is encoded as $\sum_{x_1} \dots \sum_{x_{n-1}} \mathbb{P}(\top)$. The left hand side equals

$$\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} \sum_{\hat{y}_1} \cdots \sum_{\hat{y}_n} p_{\hat{x}_1 \dots \hat{x}_n, \hat{y}_1 \dots \hat{y}_n} \sum_{y_1} \cdots \sum_{y_n} \delta_{\hat{x}_1 \dots \hat{x}_n, \hat{y}_1 \dots \hat{y}_n}(y_1, \dots, y_n)$$

according to Fact 8.14. Since the last sum ranges over all values of y_i, it counts the number of satisfying assignments to ϕ given $\hat{x}_i$. Writing this count as $\#_y(\phi(\hat{x}_1 \dots \hat{x}_n))$, the expression becomes:

$$\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} \sum_{\hat{y}_1} \cdots \sum_{\hat{y}_n} p_{\hat{x}_1 \dots \hat{x}_n, \hat{y}_1 \dots \hat{y}_n} \#_y(\phi(\hat{x}_1 \dots \hat{x}_n)) \,.$$

Since $\#_y(\phi(\hat{x}_1 \dots \hat{x}_n))$ does not depend on $\hat{y}$, we can write the expression as

$$\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \dots \hat{x}_n} \#_y(\phi(\hat{x}_1 \dots \hat{x}_n))$$

whereby $p_{\hat{x}_1 \dots \hat{x}_n} = \sum_{\hat{y}_1} \dots \sum_{\hat{y}_n} p_{\hat{x}_1 \dots \hat{x}_n, \hat{y}_1 \dots \hat{y}_n}$.

If ψ is satisfiable, there is an assignment $\hat{x}_1, \dots, \hat{x}_n$ with $\#_y(\phi(\hat{x}_1 \dots \hat{x}_n)) \geq 2^{n-1}$. If we set $p_{\hat{x}_1 \dots \hat{x}_n} = 1$ and every other probability $p_{\hat{x}'_1 \dots \hat{x}'_n} = 0$, φ is satisfied. The value of the variables Y_i can be chosen arbitrarily and in particular deterministic. The "xor-mask" described earlier ensures that the marginalization iterates over all assignments.

If φ is satisfiable, consider the assignment $x_1^{\max}, \dots, x_n^{\max}$ that maximizes the quantity $\#_y(\phi(x_1^{\max} \dots x_n^{\max}))$. Then

$$\begin{aligned}
2^{n-1} &\leq \sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \dots \hat{x}_n} \#_y(\phi(\hat{x}_1 \dots \hat{x}_n)) \\
&\leq \sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \dots \hat{x}_n} \#_y(\phi(x_1^{\max} \dots x_n^{\max})) \\
&= \left(\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \dots \hat{x}_n} \right) \#_y(\phi(x_1^{\max} \dots x_n^{\max})) \\
&= \#_y(\phi(x_1^{\max} \dots x_n^{\max})) \,.
\end{aligned}$$

Hence ψ is satisfied for $x_1^{\max} \dots x_n^{\max}$. □

To show containment in $\mathsf{NP}^{\mathsf{PP}}$, we again use a small model property.

8.17 Lemma. $\mathrm{SAT}_{\mathcal{L}_1}^{lin\langle\Sigma\rangle}$ *is in* $\mathsf{NP}^{\mathsf{PP}}$.

Proof. First we show that satisfiable instances have solutions of polynomial size.

We write each (sum of a) primitive in the arithmetic expressions as

$$\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \ldots \hat{x}_n} \sum_{x_{i_1}} \cdots \sum_{x_{i_l}} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l})$$

according to Fact 8.14.

The value of all $p_{\hat{x}_1 \ldots \hat{x}_n}$ can be encoded as a vector $\vec{p} \in \mathbb{R}^{c^n}$. For each $\hat{x_1}, \ldots, \hat{x}_n$, the sum $\sum_{x_{i_1}} \cdots \sum_{x_{i_l}} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l})$ is a constant integer $\leq c^l$. Suppose there are k such sums or primitives in the instance, whose values can be encoded as a matrix $A \in \mathbb{R}^{k \times c^n}$.

Then the value of every sum in the instance is given by the vector $A\vec{p} \in \mathbb{R}^k$.

There exists a non-negative vector $\vec{q} \in \mathbb{Q}^{c^n}$ containing at most k non-zero entries with $A\vec{q} = A\vec{p}$ (Lemma 2.5 in [FHM90]). By including a constraint $\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \ldots \hat{x}_n} = 1$ when constructing the matrix A, we can ensure that all values in $\vec{q}$ are valid probabilities. $\vec{q}$ is a polynomial sized solution with polynomial bit-sizes (Lemma 4.10 in [FHM90]) and can be guessed non-deterministically.

For this solution, the sum $\sum_{\hat{x}_1} \cdots \sum_{\hat{x}_n} p_{\hat{x}_1 \ldots \hat{x}_n}$ can then be evaluated (over all non-zero entries $p_{\hat{x}_1 \ldots \hat{x}_n}$) in polynomial time. Each sum $\sum_{x_{i_1}} \cdots \sum_{x_{i_l}} \delta_{\hat{x}_1 \ldots \hat{x}_n}(x_{i_1}, \ldots, x_{i_l})$ can be evaluated using the PP oracle because a PP oracle is equivalent to a #P oracle which can count the number of satisfying assignments of δ. □

Combining Lemmas 8.16 and 8.17 we get

8.18 Theorem. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_1}$ *and* $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_1}$ *are* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

The lower bound in Lemma 8.16 only requires a deterministic model while the upper bound in Lemma 8.17 ends up explicitly guessing a model of a specific size. Thus we directly get

8.19 Theorem. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$, $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ *are* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

8.4.2 The Interventional (Causal) Level

In causal formulas, one can perform interventions to set the value of a variable, which can recursively affect the value of all endogenous variables that depend on the intervened variable. This naturally corresponds to the choice of a variable value by an existential or universal quantifier in a Boolean formula, since in a Boolean formula with multiple quantifiers, the value chosen by each quantifier can depend on the values chosen by earlier quantifiers. Thus, an interventional equation can encode a quantified Boolean formula. This makes $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2}$ PSPACE-hard. As we will show below, it is even PSPACE-complete.

As a first tool, we see that interventional queries are the first to be able to ensure a causal ordering on the exogenous variables.

8.20 Lemma. *In $\mathcal{L}_2^*$ one can encode a causal ordering for any $* \in Lab$.*

Proof. Given a formula in $\mathcal{L}_2^*$, we add a new variable C and add to each primitive the intervention $[C{=}0]$. This does not change the satisfiability of (in-)equalities.

Given a causal order $V_{i_1} \prec V_{i_2} \prec \ldots$, we add c equations for each variable V_{i_j}, $j > 1$:

$$\mathbb{P}([C{=}1, V_{i_{j-1}}{=}k]V_{i_j}{=}k) = 1 \qquad \text{for } k = 1, \ldots, c.$$

The equations ensure, that if one variable is changed, and $C = 1$ is set, the next variable in the causal ordering has the same value, thus fixing an order from the first to the last variable. □

This insight is the crucial component in proving the PSPACE-hardness of $\mathrm{SAT}_{\mathcal{L}_2}^{base\langle\Sigma\rangle}$, it allows us to encode the order of the quantifiers.

8.21 Lemma. $\mathrm{SAT}_{\mathcal{L}_2}^{base\langle\Sigma\rangle}$ *is* PSPACE*-hard.*

Proof. We reduce from the canonical PSPACE-complete problem QBF of deciding whether a quantified Boolean formula $Q_1 x_1\, Q_2 x_2 \cdots Q_n x_n\, \psi$ with quantifiers $Q_1, \ldots, Q_n \in \{\exists, \forall\}$ is a true sentence. We introduce Boolean random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ to represent the values of the variables and denote by $\mathbf{Y} = \{Y_1, \ldots Y_k\} \subseteq \mathbf{X}$ the universally quantified variables. By 8.20 we can enforce an ordering $X_1 \prec X_2 \prec \ldots \prec X_n$ of variables, i.e. variable X_i can only depend on variables X_j with $j < i$. Our only further constraint is

$$\sum_{\mathbf{y}} \mathbb{P}([\mathbf{y}]\psi') = 2^k \tag{8.22}$$

where ψ' is obtained from ψ by replacing positive literals x_i by $X_i = 1$ and negative literals $\overline{x_i}$ by $X_i = 0$.

Suppose the constructed $\mathrm{SAT}_{\mathcal{L}_2}^{base\langle\Sigma\rangle}$ formula is satisfied by a model $\mathfrak{M}$. We show that $Q_1 x_1\, Q_2 x_2 \cdots Q_n x_n\, \psi$ is satisfiable. Each probability implicitly sums over all possible values $\mathbf{u}$ of the exogenous variables. Fix one such $\mathbf{u}$ with positive probability. Combined with $X_1 \prec \ldots \prec X_n$ this implies all random variables now deterministically depend only on the previous variables. Equation (8.22) enforces $\mathbb{P}([\mathbf{y}]\psi') = 1$ for every choice of $\mathbf{y}$ and thus simulates the $\mathbf{Y}$ being universally quantified. As the existential variables x_i, we then choose the value x_i of X_i which can only depend on X_j with $j < i$. The formula ψ' and thus ψ is then satisfied due to $\mathbb{P}([\mathbf{y}]\psi') = 1$.

On the other hand, suppose $Q_1 x_1\, Q_2 x_2 \cdots Q_n x_n\, \psi$ is satisfiable, We create a deterministic model $\mathfrak{M}$ as follows: The value of existentially quantified variables X_i is computed by the function $F_i(x_1, \ldots, x_{i-1})$ defined as the existentially chosen value when the previous

variables are set to $x_1, \dots, x_{i-1}$. The values of the universally quantified variables do not matter since we intervene on them before every occurrence. This satisfies the required order of variables and, since ψ is satisfied, we have $\mathbb{P}([\mathbf{y}]\psi') = 1$ for every choice of the universally quantified variables $\mathbf{y}$, thus satisfying equation (8.22). □

To show $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2} \in \mathsf{PSPACE}$ we give an explicit algorithm: Algorithm 1. This algorithm uses two major ideas: First, instead of guessing a model with dependencies including the exogenous variable, we consider separate models for each choice of the exogenous variable[9] The basic idea of the algorithm is that rather than guessing a model and evaluating each sum with its interventions, we enumerate all possible interventions (and resulting values) and increment each sum that includes the intervention. Thereby, rather than storing the functions and interventions, we only need to store and update the value of the sums.

[9]Remember that, by Lemma 8.3 we can assume there only to be a single exogenous variable.

8.23 Lemma. $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2}$ *is in* PSPACE.

Proof. We only need to show the correctness of Algorithm 1 and that it runs in polynomial space.

By definition, each sum $\sum_{\mathbf{y}_i} \mathbb{P}([\alpha_i]\delta_i)$ in the input can be written as a linear combination $\sum_{\mathbf{u}} p_{\mathbf{u}} \sum_{\mathbf{y}_i:\mathcal{F},\mathbf{u}\models[\alpha_i]\delta_i} 1$, where the second sum does not depend on $p_{\mathbf{u}}$. Similarly as in Lemma 8.17, these sums form a system of linear (in-)equalities in $p_{\mathbf{u}}$. By the same argument, a small model property follows that there are only polynomially many, rational probabilities $p_{\mathbf{u}}$ of polynomial bit-size. These can be guessed non-deterministically.

Next, we combine the terms $\sum_{\mathbf{u}} p_{\mathbf{u}}$ of all sums (implicitly). For example, two sums $\sum_{\mathbf{y}_i} \mathbb{P}([\alpha_i]\delta_i) + \sum_{\mathbf{y}_j} \mathbb{P}([\alpha_j]\delta_j)$ can be rewritten as

$$\begin{aligned}\sum_{\mathbf{y}_i} \mathbb{P}([\alpha_i]\delta_i) + \sum_{\mathbf{y}_j} \mathbb{P}([\alpha_j]\delta_j) &= \sum_{\mathbf{u}} p_{\mathbf{u}} \sum_{\mathbf{y}_i:\mathcal{F},\mathbf{u}\models[\alpha_i]\delta_i} 1 + \sum_{\mathbf{u}} p_{\mathbf{u}} \sum_{\mathbf{y}_j:\mathcal{F},\mathbf{u}\models[\alpha_j]\delta_j} 1 \\ &= \sum_{\mathbf{u}} p_{\mathbf{u}} \left(\sum_{\mathbf{y}_i:\mathcal{F},\mathbf{u}\models[\alpha_i]\delta_i} 1 + \sum_{\mathbf{y}_j:\mathcal{F},\mathbf{u}\models[\alpha_j]\delta_j} 1 \right).\end{aligned}$$

The algorithm performs this sum over $\mathbf{u}$ in line 5 and calculates the next sums in the subfunction. We can and will ignore the actual values $\mathbf{u}$. Relevant is only that the value of the sums is multiplied by $p_{\mathbf{u}}$ and that the functions $\mathcal{F}$ might change in each iteration.

Recall that the functions $\mathcal{F} = (F_1, \dots, F_n)$ determine the values of the endogenous variables, that is the value of variable X_i is given by $x_i = F_i(\mathbf{u}, x_1, \dots, x_{i-1})$. Thereby the functions (i.e. variables) have a causal order $X_1 \prec \dots \prec X_n$, such that the value x_i only depends on variables X_j with $j < i$. In reverse, this means that each intervention on a variable X_i can only change variables X_j with $j > i$.

Input: $\mathrm{SAT}_{\mathcal{L}_2}^{lin\langle\Sigma\rangle}$ instance
Output: Is the instance satisfiable?

Guess small model probabilities $p_{\mathbf{u}}$;
Guess a causal order $X_1, \ldots, X_n$;
Rewrite each sum $\sum_{\mathbf{y}_i} \mathbb{P}([\alpha_i]\delta_i)$ in the input as $\sum_{\mathbf{u}} p_{\mathbf{u}} \sum_{\mathbf{y}_i:\mathcal{F},\mathbf{u}\models[\alpha_i]\delta_i} 1$;
Initialize a counter c_i to zero for each such sum;
for $p_{\mathbf{u}} > 0$ **do**
    Guess values $x_1, .., x_n$;
    Simulate-Interventions$(1, \{\}, x_1, ..., x_n)$;
end
Replace the sums by c_i and verify whether the (in-)equalities are satisfied;

Function *Simulate-Interventions*$(i, \mathcal{I}, x_1, \ldots, x_n)$
    Input: Current variable X_i; set of intervened variables $\mathcal{I}$; values $x_1, \ldots, x_n$
    if $i > n$ **then**
        for *each sum counter* c_j **do**
            for *all possible values* $\mathbf{y}_j$ *of the sum* **do**
                if α_j *(after inserting* $\mathbf{y}_i$*) is* $[\{X_i{=}x_i$ *for* $i \in \mathcal{I}\}]$ *and* $x_1, .., x_n$ *satisfy* δ_j *(after inserting* $\mathbf{y}_i$*)* **then**
                    increment c_j by $p_{\mathbf{u}}$;
                end
            end
        end
    else
        Simulate-Interventions$(i + 1, \mathcal{I}, x_1, ..., x_n)$;
        for *value* v **do**
            Let $x'_1, ..., x'_n := x_1, ..., x_n$;
            $x'_i := v$;
            if $v \neq x_i$ **then**
                guess new values $x'_{i+1}, ..., x'_n$;
            end
            Simulate-Interventions$(i + 1, \mathcal{I} \cup \{i\}, x'_1, ..., x'_n)$;
        end
    end

Algorithm 1: Solving $\mathrm{SAT}_{\mathcal{L}_2}^{lin\langle\Sigma\rangle}$

Rather than storing the functions $\mathcal{F} = (F_1, \ldots, F_n)$, the algorithm only stores the values $x_i = F_i(\mathbf{u}, x_1, \ldots, x_{i-1})$, i.e. the values of the endogenous variables. The algorithm knows these values as well as the causal order, since they can be guessed non-deterministically.

The subfunction Simulate-Interventions then performs all possible interventions recursively, intervening first on variable X_1, then X_2, ..., until X_n. The parameter i means an intervention on variable X_i, $\mathcal{I}$ is the set of all variables that have been explicitly intervened on, and $x_1, \ldots, x_n$ are the current values.

In line 20, it proceeds to the next variable, without changing the current variable X_i (simulating all possible interventions includes intervening on only a subset of variables). In line 21, it enumerates all values x'_i for variable X_i. If $x_i = x'_i$, then the intervention does nothing. If $x_i \neq x'_i$, then the intervention might change all variables X_j with $j > i$ (because the function F_j might depend on X_i and change its value). This is simulated by guessing the new values x'_j. Thereby, we get the new values without considering the functions.

In the last call, line 27, it has completed a set of interventions $\mathcal{I}$. The function then searches every occurrence of the interventions $\mathcal{I}$ in the (implicitly expanded) input formula. That is, for each sum $\sum_{\mathbf{y}_j : \mathcal{F}, \mathbf{u} \models [\alpha_j]\delta_j} 1$, it counts how often $\alpha_j = \alpha$ occurs in the sum while the values x_i satisfy δ_j. [10]

[10] Rather than incrementing the counter c_j by 1 and then multiplying the final result by $p_{\mathbf{u}}$, we increment it directly by $p_{\mathbf{u}}$.

Since each intervention is enumerated only once, in the end, it obtains for all sums their exact value. It can then verify whether the values satisfy the (in-)equalities of the input.

If a satisfying model exists, the algorithm confirms it, since it can guess the probabilities and the values of the functions. In reverse, if the algorithm returns true, a satisfying model can be constructed. The probabilities directly give a probability distribution $P(\mathbf{u})$. The functions F_i can be constructed because, for each set of values $\mathbf{u}, x_1, \ldots, x_{i-1}$, only a single value for x_i is guessed, which becomes the value of the function.

Algorithm 1 runs in non-deterministic polynomial space and thus in PSPACE. □

Combining Lemmas 8.21 and 8.23 we get

8.24 Theorem. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_2}$ *and* $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2}$ *are* PSPACE*-complete.*

Since Lemma 8.21 constructs a deterministic model and Lemma 8.23 explicitly guesses the probabilities $p_{\mathbf{u}}$, it is easy to adapt both of these proofs to also show PSPACE-completeness of the corresponding small model variants:

8.25 Theorem. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{sm,\mathcal{L}_2}$ *and* $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{sm,\mathcal{L}_2}$ *are* PSPACE*-complete.*

8.4.3 The Counterfactual Level

On the counterfactual level, one can perform multiple interventions, and thus compare different functional dependencies within the model to each other. Moreover, adding in marginalizations allows to compactly combine exponentially many constraints into just a few equations. Hence, the formulas can only be evaluated if the entire – exponential-sized – deterministic part of the model is known. Thus deciding the satisfiability requires exponential time and non-determinism to find the model, making $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle}$ NEXP-complete.

The containment of $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle}$ in NEXP immediately follows from Theorem 8.9 by expanding all sums of a $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle}$ instance explicitly. This creates a $\mathrm{SAT}_{\mathcal{L}_3}^{lin}$ instance of exponential size, which can be solved non-deterministically in polynomial time in the expanded size.

8.26 Corollary. $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle} \in$ NEXP.

To prove the NEXP-hardness of $\mathrm{SAT}_{\mathcal{L}_3}^{base\langle\Sigma\rangle}$ we use the satisfiability problem of Schönfinkel-Bernays sentences. The class of Schönfinkel-Bernays sentences (also called Effectively Propositional Logic, EPR) is a fragment of first-order logic formulas where satisfiability is decidable. Each sentence in the class is of the form $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ whereby ψ can contain logical operations $\wedge, \vee, \neg$, variables $\mathbf{x}$ and $\mathbf{y}$, equalities, and relations $R_i(\mathbf{x}, \mathbf{y})$ which depend on a set of variables, but ψ cannot contain any quantifiers or functions. Determining whether a Schönfinkel-Bernays sentence is satisfiable is an NEXP-complete problem [Lew80] even if all variables are restricted to binary values [Ach15].

To simplify the notation in the proof a bit, we introduce some syntactic sugar: We will represent Boolean values as the value of the random variables, with 0 meaning FALSE and 1 meaning TRUE. We will assume that $c = 2$, so that all random variables are binary, i.e. $\mathit{Val} = \{0, 1\}$. We write (in)equalities between random variables as $=$ and $\neq$. In the binary setting, $X{=}Y$ is an abbreviation for $(X{=}0 \wedge Y{=}0) \vee (X{=}1 \wedge Y{=}1)$, and $X{\neq}Y$ is an abbreviation for $\neg(X{=}Y)$. To abbreviate interventions, we will write $[w]$ for $[W{=}w]$, $[\mathbf{w}]$ for interventions on multiple variables $[\mathbf{W}{=}\mathbf{w}]$, and $[\mathbf{v} \setminus \mathbf{w}]$ for interventions on all variables $\mathbf{V}$ except $\mathbf{W}$.

8.27 Lemma. $\mathrm{SAT}_{\mathcal{L}_3}^{base\langle\Sigma\rangle}$ *is* NEXP*-hard.*

Proof. We reduce from the satisfiability of Schönfinkel-Bernays sentences with binary variables.

We use random variables $\mathbf{X} = \{X_1, \dots X_n\}$ and $\mathbf{Y} = \{Y_1, \dots Y_n\}$ for the quantified Boolean variables $\mathbf{x}, \mathbf{y}$ in the Schönfinkel-Bernays sentence $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$. For each distinct k-ary relation $R_i(z_1, \dots, z_k)$ in the formula, we define a random variable R_i and variables $Z_i^1, \dots, Z_i^k$ for the arguments. For the j-th occurrence of that relation $R_i(t_{ij}^1, \dots, t_{ij}^k)$ with $t_{ij}^\ell \in \{x_1, \dots, x_n, y_1, \dots, y_n\}$, we define another random variable R_i^j.

We use the following constraint to ensure that R_i only depends on its arguments:

$$\sum_{\mathbf{v}} \mathbb{P}([z_i^1, \ldots, z_i^k] R_i \neq [\mathbf{v} \setminus r_i] R_i) = 0 \tag{8.28}$$

[11] All variables includes the variables $Z_i^1, \ldots, Z_i^k$.

Here, $\sum_{\mathbf{v}}$ refers to summing over all values of all endogenous variables[11] in the model and the constraint says that an intervention on $Z_i^1, \ldots, Z_i^k$ gives the same result for R_i as an intervention on $Z_i^1, \ldots, Z_i^k$ and the remaining variables, excluding R_i.

Similarly, we ensure that R_i^j only depends on its arguments:

$$\sum_{\mathbf{v}} \mathbb{P}([t_{ij}^1, \ldots, t_{ij}^k] R_i^j \neq [\mathbf{v} \setminus r_i^j] R_i^j) = 0 \tag{8.29}$$

and that R_i^j and R_i have an equal value for equal arguments:

$$\sum_{t_{ij}^1, \ldots, t_{ij}^k} \mathbb{P}([T_{ij}^1 = t_{ij}^1, \ldots, T_{ij}^k = t_{ij}^k] R_i^j \neq [Z_i^1 = t_{ij}^1, \ldots, Z_i^k = t_{ij}^k] R_i) = 0. \tag{8.30}$$

We add the following constraint for each X_i to ensure that the values of $\mathbf{X}$ are not affected by the values of $\mathbf{Y}$:

$$\sum_{\mathbf{v}} \sum_{\mathbf{y}'} \mathbb{P}([\mathbf{v} \setminus x_i] X_i \neq [\mathbf{v} \setminus (x_i, \mathbf{y}), \mathbf{Y} = \mathbf{y}'] X_i) = 0 \tag{8.31}$$

Here the outer sum sums over all values $\mathbf{v}$ of all endogenous variables $\mathbf{V}$ (including X_i and $\mathbf{Y}$), and the inner sum sums over all possible values for the variables $\mathbf{Y}$. The intervention $[\mathbf{v} \setminus x_i]$ intervenes on all variables except X_i and sets the values $\mathbf{y}$ to the values of the outer sum. The intervention $[\mathbf{v} \setminus (x_i, \mathbf{y}), \mathbf{Y} = \mathbf{y}']$ intervenes on all variables except X_i and sets the values $\mathbf{y}$ to the values $\mathbf{y}'$ of the inner sum. The constraint thus ensures that the value of X_i does not change when changing $\mathbf{Y}$ from $\mathbf{y}$ to $\mathbf{y}'$.

Let ψ' be obtained from ψ by replacing equality and relations on the Boolean values with the corresponding definitions for the random variables:

$$\sum_{\mathbf{y}} \mathbb{P}([\mathbf{y}] \psi') = 2^n \tag{8.32}$$

Suppose the $\mathrm{SAT}_{\mathcal{L}_3}^{base\langle\Sigma\rangle}$ instance is satisfied by a model $\mathfrak{M}$. We need to show $\exists \mathbf{x} \forall \mathbf{y}\, \psi$ is satisfiable. Each probability $\mathbb{P}(\ldots)$ implicitly sums over all possible values $\mathbf{u}$ of the exogenous variables. The values $\mathbf{x}$ of the variables $\mathbf{X}$ might change together with the values $\mathbf{u}$, however, any values $\mathbf{x}$ that are taken at least once can be used to satisfy $\exists \mathbf{x} \forall \mathbf{y}\, \psi$: If there was any $\mathbf{x}$ that would not satisfy ψ for all values of $\mathbf{y}$, $\mathbb{P}([\mathbf{y}] \psi')$ would be less than 1 for these values of $\mathbf{x}$ (determined by $\mathbf{u}$) and $\mathbf{y}$, and equation (8.32) would not be satisfied. From now on, fix one such $\mathbf{u}$ and the resulting values $\mathbf{x}$.

For each relation R_i, we choose the values given by the random variable R_i. Each occurrence $R_i^j(t_{ij}^1, \dots, t_{ij}^k)$ has a value that is given by $[Z_i^1{=}t_{ij}^1, \dots, Z_i^k{=}t_{ij}^k]R_i$ in the model $\mathfrak{M}$. Due to equations (8.29) and (8.30), that is the same value as $[T_i^1{=}t_{ij}^1, \dots, T_{ij}^k{=}t_{ij}^k]R_i^j$, which is the value used in $[\mathbf{y}]\psi'$. Since $[\mathbf{y}]\psi'$ is satisfied, so is ψ and $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$.

Suppose $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ is satisfiable. We create a deterministic model $\mathfrak{M}$ as follows: The values of the random variables $\mathbf{X}$ are set to the values chosen by $\exists\mathbf{x}$. The relation random variables R_i are functions depending on random variables $Z_i^1, \dots, Z_i^k$ that return the value of the relation $R_i(z_1, \dots, z_k)$. The relation random variables R_i^j on arguments $T_{ij}^1, \dots, T_{ij}^k$ return the value of the relation $R_i(t_{ij}^1, \dots, t_{ij}^k)$. This satisfies equations (8.28) and (8.29) because the functions only depend on their arguments, and equation (8.30) because the functions result from the same relation. All other random variables can be kept constant, which satisfies equation (8.31) (despite being constant in the model, the causal interventions can still change their values). Finally, equation (8.32) holds, because ψ is satisfied for all $\mathbf{Y}$. □

Combining Corollary 8.26 and Lemma 8.27 we get

8.33 Theorem. $\mathrm{SAT}_{\mathcal{L}_3}^{base\langle\Sigma\rangle}$ *and* $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle}$ *are* NEXP*-complete.*

In Corollary 8.26 we expanded the marginalization explicitly and then used Theorem 8.9 to solve the resulting instance (of now exponential size) in NP, resulting in a NEXP upper bound for $\mathrm{SAT}_{\mathcal{L}_3}^{lin\langle\Sigma\rangle}$. By using Lemma 8.12 instead of Theorem 8.9, we can use the same idea to show $\mathrm{SAT}_{sm,\mathcal{L}_3}^{lin\langle\Sigma\rangle} \in \mathsf{NEXP}$[12]. Additionally, the NEXP-hardness reduction of Lemma 8.27 always creates a deterministic model and thus also shows NEXP-hardness of $\mathrm{SAT}_{sm,\mathcal{L}_3}^{base\langle\Sigma\rangle}$. Combined:

[12] The careful reader might notice that this leaves us with a special situation: During the expansion we could even afford an exponential blowup of the small model size parameter p, allowing us to solve $\mathrm{SAT}_{\mathcal{L}_\circ}^{lin\langle\circ\rangle} \in \mathsf{NEXP}$, even when p is given in binary. This, however, is not as special as it might seem at first: For all of our problems with an inherent small model property we can always switch between the general and the small model algorithm, depending on the size of p.

8.34 Theorem. $\mathrm{SAT}_{sm,\mathcal{L}_3}^{base\langle\Sigma\rangle}$ *and* $\mathrm{SAT}_{sm,\mathcal{L}_3}^{lin\langle\Sigma\rangle}$ *are* NEXP*-complete.*

8.5 The Impact of Marginalization on Polynomial Languages

Van der Zander, Bläser and Liśkiewicz [ZBL23] prove that $\mathrm{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ and $\mathrm{SAT}_{\mathcal{L}_2}^{poly\langle\Sigma\rangle}$ are complete for succ-$\exists\mathbb{R}$ (which is the same as $\mathsf{NEXP}_{\mathrm{real}}$, see Lemma 6.5) whenever the basic terms are allowed to also contain conditional probabilities. They, however, leave open the exact complexity status of $\mathrm{SAT}_{\mathcal{L}_3}^{poly\langle\Sigma\rangle}$. In this section we both resolve this question by proving that all of the $\mathrm{SAT}_{\mathcal{L}_i}^{poly\langle\Sigma\rangle}$ are succ-$\exists\mathbb{R}$-complete and strengthen the lower bounds to no longer require conditional probabilities.

Moreover, these are the first languages that do not have the small model property (more on that in Section 8.5.2). Indeed, enforcing the small model property vastly reduces the complexity, see Table 8.3 for a summary of the results.

Terms	Probabilistic	Interventional	Counterfactual
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$		

Terms	Probabilistic	Interventional	Counterfactual
poly & marg.	$\exists\mathbb{R}^{\Sigma} = \mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$	NEXP	

Table 8.3: The first table contains the completeness results for the languages $\mathrm{SAT}_{\mathcal{L}_i}^{poly\langle\Sigma\rangle}$ (originally by [ZBL23] for $i = 1, 2$, expanded in Theorems 8.35 and 8.36). The second table contains the completeness results for the languages $\mathrm{SAT}_{sm,\mathcal{L}_i}^{poly\langle\Sigma\rangle}$ (Lemmas 8.40 and 8.41 for $i = 1$ and Theorem 8.47 for $i = 2, 3$).

8.5.1 Unconstrained Models

We start by showing that $\mathrm{SAT}_{\mathcal{L}_3}^{poly\langle\Sigma\rangle}$ is in $\mathsf{NEXP}_{\text{real}}$, with and without conditional probabilities.

8.35 Theorem. $\mathrm{SAT}_{\mathcal{L}_3}^{poly\langle\Sigma\rangle} \in \mathsf{NEXP}_{real}$. *This also holds true if we allow the basic terms to contain conditional probabilities.*

Proof. By Lemma 8.10 we have $\mathrm{SAT}_{\mathcal{L}_3}^{poly} \in \exists\mathbb{R}$ (both with and without conditional probabilities). Thus, by Lemma 6.3, $\mathrm{SAT}_{\mathcal{L}_3}^{poly}$ can be solved by a non-deterministic real word RAM in polynomial time.

As a result, a non-deterministic real word RAM for $\mathrm{SAT}_{\mathcal{L}_3}^{poly\langle\Sigma\rangle}$ can first expand all marginalizations explicitly, and then solve the resulting (exponential size) $\mathrm{SAT}_{\mathcal{L}_3}^{poly}$ instance in exponential time. □

We note that Ibeling et al. [IIM24, Theorem 3] independently obtained a variant of the above result, also using our machine characterization of succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$ given in Lemma 6.5.

To prove that $\mathrm{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ is succ-$\exists\mathbb{R}$-complete, van der Zander et al. [ZBL23] show the hardness part for the variant of the probabilistic language where the primitives are also allowed to be conditional probabilities. A novel contribution of our work is to extend this completeness result to our version for languages which disallow conditional probabilities:

8.36 Theorem. *The problem* $\mathrm{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ *remains* succ-$\exists\mathbb{R}$*-complete even without conditional probabilities.*

In the rest of this section, we will give the proof of the theorem. In their hardness proof, [ZBL23] use and show the succ-$\exists\mathbb{R}$-completeness of Σ_{vi}-ETR (see Chapter 6 for more info). We define the intermediate problem Σ_{vi}-ETR$_1$ in the same way as Σ_{vi}-ETR, but asking the question whether there is a solution where the sum of the absolute values (ℓ_1 norm) is bounded by 1. Then we can reduce Σ_{vi}-ETR$_1$ to $\mathrm{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ without the need for conditional probabilities.

To reduce Σ_{vi}-ETR to Σ_{vi}-ETR$_1$, we use a result by Grigoriev and Vorobjov [GV88] who showed that the solution to an ETR instance can be bounded by a constant that only depends on the bit-size of the instance.

8.37 Theorem (Grigoriev and Vorobjov [GV88])**.** *Let $f_1, \dots, f_k \in \mathbb{R}[X_1, \dots, X_n]$ be polynomials of total degree $\leq d$ with coefficients of bit size $\leq L$. Then every connected component of $\{x \in \mathbb{R}^n \mid f_1(x) \geq 0 \wedge \dots \wedge f_k(x) \geq 0\}$ contains a point of distance less than $2^{Ld^{cn}}$ from the origin for some absolute constant c. The same is true if some of the inequalities are replaced by strict inequalities.*

Knowing a bound on the size of our solutions allows us to scale down the solutions to our desired size.

8.38 Lemma. Σ_{vi}-ETR $\leq_P \Sigma_{vi}$-ETR$_1$.

Proof. Let ϕ be an instance of Σ_{vi}-ETR. We will transform it into a formula φ such that φ has a solution with ℓ_1 norm bounded by 1 iff ϕ has any solution.

Let S be the bit length of ϕ. The number n of variables in ϕ is bounded by 2^S. The degree of all polynomials is bounded by S. Note that the exponential sums do not increase the degree at all. Finally, all coefficients have bit size $O(S)$. Note that one summation operator doubles the coefficients at most. By Theorem 8.37, if ϕ is satisfiable, there is a solution with entries bounded by $2^{O(S)\cdot S^{c\cdot 2^S}}$, which is bounded by $T := 2^{2^{2^{cS}}}$ for some constant c.

In our new instance φ first create a small constant $d \leq 1/((1+2^m+n)T)$ for some m polynomial in S defined below. This can be done using Tseitin's trick: We take 2^m+1 many fresh variables t_i and start with $(1+2^m+2^S)t_0 = 1$ and then iterate by adding the equation $\sum_{i=0}^{2^m-1}(t_i^2 - t_{i+1})^2 = 0$, i.e. forcing $t_{i+1} = t_i^2$. To implement the first equation we replace 2^m by $\sum_{e_1=0}^{1}\cdots\sum_{e_m=0}^{1} 1$ and similarly replace 2^S. To implement the second equation we replace $\sum_{i=0}^{2^m-1}(t_i^2 - t_{i+1})^2$ by $\sum_{e_1=0}^{1}\cdots\sum_{e_m=0}^{1}\sum_{f_1=0}^{1}\cdots\sum_{f_m=0}^{1}(t_{e_1,\dots,e_m}^2 - t_{f_1,\dots,f_m})^2 \cdot A(e_1,\dots,e_m,f_1,\dots,f_m)$ where A is an arithmetic formula returning 1 iff the binary number represented by $f_1,\dots,f_m$ is the successor of the binary number represented by $e_1,\dots,e_m$. The unique satisfying assignment to the t_i has its entries bounded by $1/(1+2^m+2^S)^{2^i} \leq 1/(1+2^m+n)^{2^i}$. Let $d := t_{2^m}$ be the last variable. The number m to reach $d \leq 1/((1+2^m+n)T)$ is polynomial in S.

Now in ϕ we replace every occurrence of x_i by x_i/d and then multiple each (in-)equality by an appropriate power of d to remove the divisions in order to obtain φ. In this way,

from every solution to ϕ, we obtain a solution to φ by multiplying the entries by d and vice versa. Whenever ϕ has a solution, then it has one with entries bounded by T. By construction φ then has a solution with entries bounded by $1/(1+2^m+n)$. Since each entry of the solution is bounded by $1/(1+2^m+n)$ and there are $1+2^m+n$ variables, the ℓ_1 norm is bounded by 1. □

This intermediate problem allows us to represent the Σ_{vi}-ETR-formula exclusively using disjoint events.

8.39 Lemma. $\Sigma_{vi}\text{-ETR}_1 \leq_P \text{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$.

Proof. Let X_0 be a random variable with range $\{-1,0,1\}$ and let $X_1,\dots,X_N$ be binary random variables[13]. We replace each real variable $x_{e_1,\dots,e_N}$ in the Σ_{vi}-ETR$_1$ formula as follows:

$$x_{e_1,\dots,e_N} := \mathbb{P}(X_0{=}1 \wedge X_1{=}e_1 \wedge \dots \wedge X_N{=}e_N) - \mathbb{P}(X_0{=}-1 \wedge X_1{=}e_1 \wedge \dots \wedge X_N{=}e_N)$$

This guarantees that $x_{e_1,\dots,e_N} \in [-1,1]$. The existential quantifiers now directly correspond to the existence of a probability distribution $P(X_0,\dots,X_N)$, where each variable corresponds to a different set of two entries of P.

[13] While we technically don't allow differing domains between variables, we can simply add constraints of the form $\mathbb{P}(X_i{=}0)+\mathbb{P}(X_i{=}1) = 1$ to create binary random variables anyways.

Let $P(X_0,\dots,X_N)$ be a solution to the constructed $\text{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ instance. Then clearly setting $x_{e_1,\dots,e_N} = P(1,e_1,\dots,e_N) - P(-1,e_1,\dots,e_N)$ satisfies the original Σ_{vi}-ETR$_1$ instance. Furthermore it has an ℓ_1 norm bounded by 1:

$$\begin{aligned}\sum_{e_1=0}^{1}\cdots\sum_{e_N=0}^{1}|x_{e_1,\dots,e_N}| &= \sum_{e_1=0}^{1}\cdots\sum_{e_N=0}^{1}|P(1,e_1,\dots,e_N)-P(-1,e_1,\dots,e_N)|\\ &\leq \sum_{e_1=0}^{1}\cdots\sum_{e_N=0}^{1}(P(1,e_1,\dots,e_N)+P(-1,e_1,\dots,e_N))\\ &\leq 1\,.\end{aligned}$$

Vice-versa, let the original Σ_{vi}-ETR$_1$ be satisfied by some choice of the $x_{e_1,\dots,e_N}$ with ℓ_1 norm α bounded by 1. We define the probability distribution

$$P(X_0,X_1,\dots,X_N) = \begin{cases}\frac{1-\alpha}{2^N} & \text{if } X_0 = 0\\ \max(x_{X_1,\dots,X_N},0) & \text{if } X_0 = 1\\ \max(-x_{X_1,\dots,X_N},0) & \text{if } X_0 = -1\end{cases}$$

Every entry of P is non-negative since $\alpha \leq 1$. Furthermore the sum of all entries is exactly 1, the entries with $X_0 \in \{-1,1\}$ contribute exactly α total and the 2^N entries with $X_0 = 0$ contribute $1-\alpha$ total. Since P fulfills the equation $x_{e_1,\dots,e_N} = P(1,e_1,\dots,e_N) - P(-1,e_1,\dots,e_N)$, it is a solution to the constructed $\text{SAT}_{\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ instance. □

Combining both of these reductions with the $\mathsf{NEXP}_{\text{real}}$-hardness of Σ_{vi}-ETR completes the proof of Theorem 8.36.

8.5.2 Small Models

In this section, we employ the satisfiability problems for languages of the causal hierarchy. The problem $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ is defined like $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathcal{L}_1}$, but in addition we require that a satisfying distribution has only polynomially large support, that is, only polynomially many entries in the exponentially large table of probabilities are nonzero. Formally we can achieve this by extending an instance with an additional unary input $p \in \mathbb{N}$ and requiring that the satisfying distribution has a support of size at most p. As we have seen in the previous sections, the upper bounds for $\mathrm{SAT}^{*}_{\mathcal{L}_i}$ for $i = 1, 2, 3$ and $* \in \{base, base\langle\Sigma\rangle, lin, lin\langle\Sigma\rangle, poly\}$ crucially rely on the fact that the considered formulas have the small model property: If the instance is satisfiable, then it is satisfiable by a small model. For $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathcal{L}_1}$, this does not seem to be true because we can directly force any model to be arbitrarily large, e.g., by enforcing a uniform distribution using $\sum_{x_1} \dots \sum_{x_n} (P(X_1{=}x_1, \dots, X_n{=}x_n) - P(X_1{=}0, \dots, X_n{=}0))^2 = 0$ Thus, we have to explicitly require that the models are small, yielding the problem $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$.

It turns out that $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ is a natural complete problem for the class $\exists\mathbb{R}^{\Sigma}$ we introduced in Section 6.6.1. We start by showing that $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ is contained in $\exists\mathbb{R}^{\Sigma}$.

8.40 Lemma. $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1} \in \exists\mathbb{R}^{\Sigma}$.

Proof. Given a $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ instance φ, we first want to ensure that all primitives are of the form $\mathbb{P}(X_1{=}\xi_1, \dots, X_N{=}\xi_n)$, that is, every variable occurs in the primitive and everything is connected by conjunctions.

We proceed inductively, so let $\mathbb{P}(\delta)$ be given. We define the arithmetization $A_\delta(\xi_1, \dots, \xi_n)$ of δ to be 1, if $\mathbb{P}(X_1{=}\xi_1, \dots, X_N{=}\xi_n)$ contributes to $\mathbb{P}(\delta)$, and 0 otherwise. Note that this is similar to Fact 8.14, however we require $A_\delta(\xi_1, \dots, \xi_n)$ to be a polynomial in $\xi_1, \dots, \xi_n$. A_δ can be easily derived from δ by following the logical structure of δ. We first present the construction for the Boolean domain for simplicity.

- In the base case δ equals $X_i = \xi$. The corresponding arithmetization is $1-(X_i-\xi)^2$.
- If $\delta = \delta_1 \wedge \delta_2$, then $A_\delta = A_{\delta_1} A_{\delta_2}$.
- If $\delta = \neg\delta_1$, then $A_\delta = 1 - A_{\delta_1}$

Since we implement A_δ as a polynomial, it is also easy to implement the starting predicates "$X_i = \xi$" over larger domains D. It is simply a polynomial of degree $|D|$ that is 1 on ξ and 0 everywhere else. Then we have

$$\mathbb{P}(\delta) = \sum_{\xi_1} \cdots \sum_{\xi_n} A_\delta(\xi_1, \dots, \xi_n) \mathbb{P}(X_1{=}\xi_1, \dots, X_n{=}\xi_n)$$

For the reduction, we now want to replace the primitives $\mathbb{P}(X_1{=}\xi_1, \dots, X_N{=}\xi_n)$ by variables. There are exponentially many primitives, but we only have polynomially

many variables. However, we are only interested in small models, so we only need to simulate exponentially many variables with small support, say of size p. The variables $X_1, \ldots, X_p$, will store the values. Selector variables $e_{i,j}$, $1 \leq i \leq p$, $1 \leq j \leq n$, will assign them to places in our big array. We quantify as follows:

$$\exists X_1, \ldots, X_p \, \exists e_{1,1}, \ldots, e_{p,n} \bigwedge_{i,j} \prod_{d \in D} (e_{i,j} - d) = 0 \ldots$$

This constraints the $e_{i,j}$ to have values in D. Then the expression

$$X(f_1, \ldots, f_n) = \sum_{i=1}^{p} X_i \prod_{j=1}^{n} (1 - (f_j - e_{i,j})^2)$$

with $f_i \in D$ simulate exponentially many variables with support size p. We can replace the $\mathbb{P}(X_1{=}\xi_1, \ldots, X_n{=}\xi_n)$ by $X(\xi_1, \ldots, \xi_n)$ and get that $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ is in $\exists\mathbb{R}^{\Sigma}$.

For larger domains D, we just have to replace the expressions $1-(X_i-\xi)^2$ or $1-(f_j-e_{i,j})^2$ by a polynomial, that is exactly 1 on the element ξ or $e_{i,j}$ and 0 on all other elements of D. Such a polynomial can be easily found using Lagrange interpolation. □

The second half of the completeness result for $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ is given in the following lemma.

8.41 Lemma. $\Sigma\text{-ETR} \leq_P \mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$.

Like before, let $\Sigma\text{-ETR}_{1/2}$ denote the following problem: Instances are the same as for Σ-ETR. However, we ask the question whether there is a solution with the sum of the absolute values (ℓ_1 norm) being bounded by $1/2$. We prove that the general Σ-ETR can be reduced to this problem.

Using this intermediate problem we prove $\Sigma\text{-ETR} \leq_P \mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$, i.e. Lemma 8.41, in two steps.

8.42 Lemma. $\Sigma\text{-ETR} \leq_P \Sigma\text{-ETR}_{1/2}$.

The proof of the first lemma is almost identical to the proof of Lemma 8.38 and therefore omitted.

8.43 Lemma. $\Sigma\text{-ETR}_{1/2} \leq_P \mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$.

Proof. Assume that our $\Sigma\text{-ETR}_{1/2}$ instance φ uses variables $x_1, \ldots, x_n$ and summation variables $e_1, \ldots, e_m$.

In the $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ instance, we will have binary random variables $X_1, \ldots, X_n$, $\overline{X_1}, \ldots, \overline{X_n}$ and E with domain $\{0, 1\}$. Variables x_i will be represented by the term

$$\begin{aligned} p_i := &\mathbb{P}(X_1{=}0, \ldots, X_i{=}1, \ldots, X_n{=}0, \overline{X_1}{=}0, \ldots, \overline{X_n}{=}0, E{=}0) \\ &- \mathbb{P}(X_1{=}0, \ldots, X_n{=}0, \overline{X_1}{=}0, \ldots, \overline{X_i}{=}1, \ldots, \overline{X_n}{=}0, E{=}0)\,. \end{aligned}$$

Note that each atomic term has exactly one variable set to 1 (either X_i or $\overline{X_i}$, representing the positive and negative components of p_i respectively) and all others to 0. Summation variables e_i will be represented using the term

$$q_{e_i} := \mathbb{P}(X_1{=}0, \ldots, X_n{=}0, \overline{X_1}{=}0, \ldots, \overline{X_n}{=}0, E{=}e_i)\,.$$

First, we enforce $q_0 = 0$ and $q_1 = 1/2$. (We use fractions for convenience. We can always multiply by the common denominator.) This places total mass $1/2$ on one entry in the distribution. Now a summation $\sum_{e_i=0}^{1}$ in φ can be simulated by the summation operator $\sum_{e_i=0}^{1}$ of $\mathrm{SAT}_{sm,\mathcal{L}_1}^{poly\langle\Sigma\rangle}$. Inside the summation, the term $2 \cdot q_{e_i}$ simulates the term e_i in the summation of φ.

The variables x_i will be simulated by the expressions p_i. Now we can simply translate φ step by step into a $\mathrm{SAT}_{sm,\mathcal{L}_1}^{poly\langle\Sigma\rangle}$ instance using the transformation above. If the new instance is satisfiable, then it has a model with support $\leq n+2$: Let α denote the slack of $\varphi \in \Sigma\text{-ETR}_{1/2}$, i.e. $1/2$ minus the ℓ_1 norm of some solution bounded by $1/2$. By choosing

$$\mathbb{P}(X_1{=}0, \ldots, X_i{=}1, \ldots, X_n{=}0, \overline{X_1}{=}0, \ldots, \overline{X_n}{=}0, E{=}0) = \min(0, x_i) + \frac{\alpha}{2n}$$

and

$$\mathbb{P}(X_1{=}0, \ldots, X_n{=}0, \overline{X_1}{=}0, \ldots, \overline{X_i}{=}1, \ldots, \overline{X_n}{=}0, E{=}0) = \min(0, -x_i) + \frac{\alpha}{2n},$$

exactly $1/2$ of the probability mass are placed on the p_i in total. Vice-versa to express any solution with ℓ_1 norm bigger than $1/2$ would require more than $1/2$ probability mass being put on the p_i, in turn making it not a valid probability distribution when added to the $1/2$ probability mass on q_1. □

Note that technically the previous proof shows a different small model property, where *sm* means

$$\#\{(x_1, \ldots, x_m) : \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) > 0\}$$

is small, i.e., is polynomially bounded in the input size. This however, implies Definition 8.1 where

$$\#\{(u_1, \ldots, u_m) : P(U_1 = u_1, \ldots, U_m = u_m) > 0\}$$

is small (we cannot generally use this alternative definition, as it is not meaningful to give constraints on the distribution $\mathbb{P}(\mathbf{X})$ for *causal* models where interventions can change this distribution in various ways).

8.44 Fact. For a given model $\mathfrak{M}$, let $\#^+_{\mathbf{X}}(\mathfrak{M}) = \#\{(x_1, \ldots, x_n) : \mathbb{P}(X_1{=}x_1, \ldots, X_n{=}x_n) > 0\}$ be the number of assignments to the observed variables with positive probability and $\#^+_{\mathbf{U}}(\mathfrak{M}) = \#\{(u_1, \ldots, u_m) : P(U_1 = u_1, \ldots, U_m = u_m) > 0\}$ be the number of assignments to the unobserved variables with positive probability.

Then $\#^+_{\mathbf{X}}(\mathfrak{M}) \leq \#^+_{\mathbf{U}}(\mathfrak{M})$.

Proof. The functions $\mathcal{F}$ map an unobserved assignment $\mathbf{u}$ to an observed assignment $\mathbf{x}$ deterministically. Thus any assignment $\mathbf{u}$ with positive probability $P(\mathbf{u})$ can only make at most a single value $\mathbb{P}(\mathbf{x})$ non-zero. □

The reverse does not hold. All functions in the model might be constant, in which case $\#^+_{\mathbf{X}} = 1$, regardless of $\#^+_{\mathbf{U}}$. Still:

8.45 Fact. For a given probabilistic formula $\varphi \in \mathcal{L}_1^{poly\langle\Sigma\rangle}$ and a number $p \in \mathbb{N}$, there exists a model $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ such that $\mathfrak{M} \models \varphi$ and $\#^+_{\mathbf{U}}(\mathfrak{M}) \leq p$ if and only if there exists a model $\mathfrak{M}' = (\mathcal{F}', P', \mathbf{U}', \mathbf{X})$ such that $\mathfrak{M}' \models \varphi$ and $\#^+_{\mathbf{X}}(\mathfrak{M}') \leq p$.

Proof. "⇒": If there exists a model $\mathfrak{M}$, we can set $\mathfrak{M}' = \mathfrak{M}$ due to Fact 8.44.

"⇐": If there exists a model $\mathfrak{M}'$, we define the model $\mathfrak{M}$ as follows:

- $\mathbf{U}$ consists of a single variable U, with its domain being the cartesian product of the domains of all $X_i \in \mathbf{X}$,
- $F_i((u_1, \ldots, u_n)) = u_i$,
- $P(U{=}(u_1, \ldots, u_n)) = \mathbb{P}'(X_1{=}u_1, \ldots, X_n{=}u_n)$.

Then $\mathbb{P} = \mathbb{P}'$ due to identifying the components of U with X_i. Thus $\mathfrak{M}$ and $\mathfrak{M}'$ are indistinguishable for all probabilistic formulas. □

Thus, the proof of Lemma 8.43 indeed proves $\exists\mathbb{R}^\Sigma$-hardness for for $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$. Combined with Lemma 8.40 we get:

8.46 Theorem. $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_1}$ *is* $\exists\mathbb{R}^\Sigma$*-complete.*

Starting with the second (interventional) level, however, the complexity increases to NEXP-completeness. Note that this is likely still weaker than the $\mathsf{NEXP}_{\mathrm{real}}$-completeness of $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathcal{L}_2}$ and $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathcal{L}_3}$.

8.47 Theorem. $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_2}$ *and* $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_3}$ *are* NEXP*-complete.*

Proof. We first show that $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_2}$ is NEXP-hard. This is done similarly as in the proof of Lemma 8.27, again reducing from the satisfiability of Schönfinkel-Bernays sentences $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$. The setup of all random variables is identical, however, this time we have to use the polynomial arithmetic to encode our constraints.

Using Lemma 8.20, we require that each model has the variable ordering $X_1 \prec \ldots \prec X_n \prec Y_1 \prec \ldots \prec Y_n$ and then the variables R_i and R_i^j in some arbitrary, but fixed order. This allows to enforce a deterministic model by adding the constraint $\sum_{\mathbf{t}} \sum_{\mathbf{v}} (\mathbb{P}([\mathbf{t}]\mathbf{v})^2 - \mathbb{P}([\mathbf{t}]\mathbf{v}))^2 = 0$ for each subset T of variables containing a (possibly empty) consecutive subset of variables starting at X_1 in our variable ordering. This enforces every probability

in this model to be either 0 or 1, and it further enforces that no variable depends on any exogenous variables but only on the endogenous variables. This determinism allows us to rephrase the additional equations using purely interventional terms:

We use the following constraint to ensure that R_i only depends on its arguments:

$$\sum_{\mathbf{v}} (\mathbb{P}([z_i^1, \dots, z_i^k]R_i{=}r_i) - \mathbb{P}([\mathbf{v} \setminus r_i]R_i{=}r_i))^2 = 0 \tag{8.48}$$

Here $\sum_{\mathbf{v}}$ refers to summing over all values of all endogenous variables in the model and the constraint says that an intervention on $Z_i^1, \dots, Z_i^k$ gives the same result for R_i as an intervention on $Z_i^1, \dots, Z_i^k$ and the remaining variables, excluding R_i.

We use the following constraint to ensure that R_i^j only depends on its arguments:

$$\sum_{\mathbf{v}} (\mathbb{P}([t_1, \dots, t_k]R_i^j{=}r_i^j) - \mathbb{P}([\mathbf{v} \setminus r_i^j]R_i^j{=}r_i^j))^2 = 0 \tag{8.49}$$

and that R_i^j and R_i have an equal value for equal arguments:

$$\sum_{t_1,\dots,t_k} (\mathbb{P}([T_1{=}t_1, \dots, T_k{=}t_k]R_i^j{=}r_i) - \mathbb{P}([Z_i^1{=}t_1, \dots, Z_i^k{=}t_k]R_i{=}r_i))^2 = 0\,. \tag{8.50}$$

We don't need to add any further constraints to ensure that the values of $\mathbf{X}$ are not affected by the values of $\mathbf{Y}$, as this is already taken care of using our variable ordering.

Let ψ' be obtained from ψ by replacing equality and relations on the Boolean values with the corresponding definitions for the random variables:

$$\sum_{\mathbf{y}} \mathbb{P}([\mathbf{y}]\psi') = 2^n \tag{8.51}$$

The correctness now follows by the same argument as in Lemma 8.27. On one hand, the model $\mathfrak{M}$ constructed there if $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ is satisfiable is already deterministic and thus fulfills all the above constraints. On the other hand, if there is a model $\mathfrak{M}$ satisfying the above constraints, all possible values $\mathbf{u}$ of the exogenous variables result in the exact same probabilities which, following the original proof, lead to $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ being satisfiable.

Furthermore $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_3} \in \mathsf{NEXP}$. We adapt the trick from Lemma 8.23. Instead of viewing the model as changing with $\mathbf{u}$, we instead consider for each value $\mathbf{u}$ a separate (deterministic) model. Due to the small-model property, we then only have to consider polynomially many models and each model has a straight-forward representation using exponentially many bits, directly encoding all the functions in $\mathcal{F}$ as explicit tables. Algorithm 2 uses this observation to solve $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{sm,\mathcal{L}_3}$ non-deterministically in exponential time. It constructs an ETR-formula with polynomially many variables, polynomially

many polynomials of polynomial degree, but potentially exponentially many monomials. Each coefficient of the formula can be represented using a polynomial bit length. Renegar's algorithm (Lemma 2.9) can thus solve the ETR-formula deterministically in time $O(\text{poly}(|\varphi|)^{O(p)})$, which is exponential in the input length. □

Input: $\mathcal{L}_3^{poly\langle\Sigma\rangle}$ formula φ and a unary number $p \in \mathbb{N}$
Output: Is the instance satisfiable with a model of size at most p?
1 Guess a causal order $X_1, \ldots, X_n$;
2 Explicitly expand all exponential sums in φ;
3 For each probability $\mathbb{P}(\delta_i)$ occurring in φ introduce a variable P_i, initialized to 0;
4 Introduce symbolic variables $z_1, \ldots, z_p$;
5 **for** $i \in \{1, \ldots, p\}$ **do**
6 | Guess a deterministic SCM $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U} = \emptyset, \mathbf{X})$ using the causal order $X_1, \ldots, X_n$;
7 | **for** *each* $\mathbb{P}(\delta_j)$ *with* $\mathcal{F} \models \delta_j$ **do**
8 | | Symbolically increment P_j by z_i.
9 | **end**
10 **end**
11 Replace all probabilities $\mathbb{P}(\delta_i)$ by the values of P_i and check whether the resulting ETR-formula (in the variables $z_1, \ldots, z_p$) is satisfiable using Renegar's algorithm;

Algorithm 2: Solving $\text{SAT}_{sm,\mathcal{L}_3}^{poly\langle\Sigma\rangle}$ in NEXP

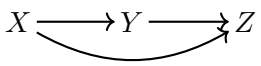
This follows because a causal model consists of a probabilistic part P and a deterministic, causal part $\mathcal{F}$. In the small model, P is restricted to have polynomial size, but $\mathcal{F}$ can still have exponential size. In $\text{SAT}_{sm,\mathcal{L}_1}^{poly\langle\Sigma\rangle}$, one cannot reason about $\mathcal{F}$, but $\text{SAT}_{sm,\mathcal{L}_2}^{poly\langle\Sigma\rangle}$ and $\text{SAT}_{sm,\mathcal{L}_3}^{poly\langle\Sigma\rangle}$ can access the full power of $\mathcal{F}$. Since NEXP subsumes $\exists\mathbb{R}$, reasoning about polynomially many real numbers (probabilities) cannot increase the complexity.

8.6 Constraining the Graph Structure

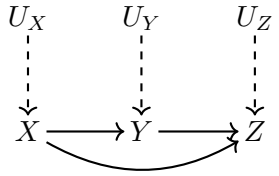
So far we have looked at SCMs with arbitrary dependencies. However, in a lot of cases during causal reasoning a researcher might already know (or have a guess at) the parents of each endogenous variable. This motivates the introduction of the problems we will be considering throughout this section: Given a DAG G and some formula $\varphi \in \mathcal{L}_i^*$, is there an SCM $\mathfrak{M}$ with the dependency structure given by G and $\mathfrak{M} \models \varphi$?

Specifying this graph in full requires knowledge about the exogenous variables and which exogenous variables influence which endogenous variables. In general this is a difficult question; by their nature, exogenous variables are unobserved.

Thus we will also consider two special cases. The first special case focuses on Markovian models. Here, every endogenous variable has their own independent exogenous variable. As an example, consider the following DAG G:



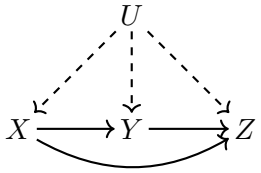
Interpreted as a Markovian model, this is considered shorthand for the following fully specified graph structure:



Further motivating this interpretation, in the purely probabilistic case, an SCM $\mathfrak{M}$ with this graph structure is equivalent to a Bayesian network[14] with graph structure G. We will see more of this correspondence in the proof of Proposition 8.68.

[14] A Bayesian network represents a probability distribution as conditional probability tables of each variable, given the values of its parents according to its graph structure G.

The second special case is motivated by going the opposite direction and follows the idea of "the exogenous variables are unobserved, so we should not assume anything about them or their relationships". By our arguments in Lemma 8.3 this implies that we can always restrict ourselves to a single exogenous variable that can influence everything. Going back to our previous example, interpreted as a semi-Markovian model, the DAG G is considered shorthand for the following fully specified graph structure:



Remarkably, the complexity between these two special cases is vastly different. On one hand, the satisfiability problems with a semi-Markovian interpretation mostly inherit the complexity of the corresponding satisfiability problem with no constraints on the model, with one notable exception: $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ are NEXP-complete, an increase in complexity from the PSPACE-completeness of $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_2}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2}$. On the other hand, the satisfiability problems with a Markovian interpretation exhibit a complexity jump over many of their unconstrained counterparts, with already $\mathrm{SAT}^{base}_{\mathrm{Markov},\mathcal{L}_1}$ being NEXP-complete. This same complexity jump is also mirrored for the general case where the input is a fully specified graph structure, however there are no further complexity increases on top of the Markovian interpretation. See Tables 8.4 and 8.5 for an overview of the completeness results we show throughout this section.

Terms	Probabilistic	Interventional	Counterfactual
basic	NEXP (a)		
lin			
poly			
basic & marg.	NEXP (a)		
lin & marg.			
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$ (b)		

Table 8.4: Completeness results for the satisfiability problems with given graph structure for the model (Markovian interpretation or fully specified graph structure). (a) Theorem 8.67, (b) Theorem 8.69

Terms	Probabilistic	Interventional	Counterfactual
basic	NP (a)		
lin			
poly	$\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$ (a)		
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$ (a)	NEXP (a+b)	
lin & marg.			
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$ (a)		

Table 8.5: Completeness results for the satisfiability problems with given graph structure for the model (semi-Markovian interpretation). (a) Corollary 8.72, (b) Theorem 8.73

8.6.1 Markovian Models

We first focus on the case of Markovian models. The crucial property for most upper bounds in the previous sections was the small model property. However small models are only ever possible if the number of exogenous variables with a domain size of at least two is very small: due to our assumption that all exogenous variables are independent, even with just $m = \log^2(n)$ binary non-degenerate exogenous variables we already have $\#\{(u_1, \dots, u_m) : P(U_1{=}u_1, \dots, U_m{=}u_m) > 0\} = n^{\log(n)}$ and thus a super-polynomially sized model. For Markovian models we have $m = n$ which implies an exponentially sized model. However, the entries of the joint probability distribution – when interpreted as an exponentially large table – are not independent of each other. They naturally decompose as the product of the distributions of the individual exogenous variables:

$$P(U_1{=}u_1, \dots, U_m{=}u_m) = \prod_{i=1}^{m} P(U_i{=}u_i)$$

This motivates the relaxed notion of *locally small models*, where we only require the degrees of freedom, or more explicitly, the size of the domains of the individual exogenous variables, to be small.

8.52 Definition (Locally Small Models)**.** The decision problems $\mathrm{SAT}^{*}_{lsm,\mathcal{L}_i}$, with $i = 1, 2, 3$, take as input a formula $\varphi \in \mathcal{L}^{*}_{i}$ and a unary encoded number $p \in \mathbb{N}$ and ask whether there exists a model $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U} = \{U_1, \dots, U_m\}, \mathbf{X})$ such that $\mathfrak{M} \models \varphi$ and $\sum_{j=1}^{m} \#\{u_j : P(U_j = u_j) > 0\} \leq p$.

The variants with added constraints on the graph structure are defined in the natural way.

With a locally small model, the "biggest" part of the description of a model is the functional dependencies. Indeed, all satisfiability problems that exhibit a local small model property[15], can be solved in NEXP by guessing the functional dependencies and using Renegar's algorithm to check whether the resulting formula (in the $P(U_i{=}u_i)$ as variables) is satisfied.

[15] We would like to remind that in all our models (not just the Markovian ones) we always assume all exogenous variables to be independent of each other. This is crucial for the local small model property to be sensible.

8.53 Lemma. $\mathrm{SAT}^{*}_{\mathrm{DAG},lsm,\mathcal{L}_i} \in \mathsf{NEXP}$ *for any* $i = 1, 2, 3$ *and* $* \in Lab$.

Proof. The claim is achieved by Algorithm 3.

Assume the ETR-formula constructed by Algorithm 3 is satisfiable. Then, the SCM $\mathfrak{M}$ constructed in the algorithm, together with $P(U_1{=}u_1, \dots, U_m{=}u_m) = z_{1,u_1} \cdot \ldots \cdot z_{n,u_n}$ follows the graph structure G and clearly has a locally small model of size at most p. Furthermore we have $\mathbb{P}(\delta_i) = \sum_{\mathbf{u}:\mathbf{u}\models\delta_i} P(\mathbf{u})$ by construction, so $\mathfrak{M} \models \varphi$. Using the same correspondence we see that these are all equivalences.

Remains to consider the running time of the algorithm: The functional dependencies are tables of exponential size and can be guessed in non-deterministic exponential time.

Input: $\mathcal{L}_i^*$ formula φ, a DAG G and a unary number $p \in \mathbb{N}$
Output: Is the instance satisfiable by a model $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U} = (U_1, \ldots, U_m), \mathbf{X})$ with fully specified graph structure G and $\sum_{j=1}^{m} \#\{u_j : P(U_j = u_j) > 0\} \leq p$?

1 Explicitly expand all exponential sums in φ;
2 Guess the functional dependencies $\mathcal{F}$ of an SCM $\mathfrak{M}$ abiding the graph structure G where the domains of the exogenous variables $\mathbf{U}$ are of total size at most p;
3 For each probability $\mathbb{P}(\delta_i)$ occurring in φ introduce a variable P_i, initialized to 0;
4 Introduce symbolic variables $z_{j,k}$ for $1 \leq j \leq m$ and each possible value k of U_j;
5 **for** *each assignment* $\mathbf{u}$ *to the exogenous variables* **do**
6 **for** *each* $\mathbb{P}(\delta_i)$ *with* $\mathfrak{M}, \mathbf{u} \models \delta_i$ **do**
7 Symbolically increment P_i by $z_{1,u_1} \cdot \ldots \cdot z_{m,u_m}$;
8 **end**
9 **end**
10 Construct an ETR-formula (in the variables $z_{j,k}$) as follows: Replace all probabilities $\mathbb{P}(\delta_i)$ by the values of P_i. Add constraints for each $z_{j,k}$ to be non-negative and $\sum_k z_{j,k} = 1$ for each j;
11 Then check whether the resulting ETR-formula is satisfiable using Renegar's algorithm;

Algorithm 3: Solving $\mathrm{SAT}^*_{\mathrm{DAG},lsm,\mathcal{L}_i}$ in NEXP

The constructed ETR formula has exponential size, however it has at most p variables. Renegar's algorithm (Lemma 2.9) can thus solve the satisfiability of that ETR-formula in exponential time. □

Notably, Lemma 8.53 works for any fully specified graph structure, not just Markovian ones. Indeed, all of the upper bounds in this section work for the general case, while we only use Markovian models for lower bounds.

To show all the NEXP upper bounds of Table 8.4, it remains to show that $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{DAG},\mathcal{L}_3}$ and $\mathrm{SAT}^{poly}_{\mathrm{DAG},\mathcal{L}_3}$ both admit locally small models. The main ingredient in proving this is the following technical lemma about systems of equations of set-multilinear polynomials, enabling us to restrict the domains of the exogenous variables. It's a generalization of Lemma 2.5 in [FHM90], the main ingredient in showing small model properties before (see Theorem 8.4 and Lemmas 8.5, 8.17 and 8.23).

8.54 Lemma. *Let $X_i = \{x_{i,1}, \ldots, x_{i,\kappa_i}\}$ for $i = 1, \ldots, n$ be sets of real non-negative variables. Furthermore let $p_j(X_1, \ldots, X_n)$ for $j = 1, \ldots, m$ be polynomials, such that no monomial contains two variables from the same set[16] X_i. Then, if there is a solution to the system of equations $p_j(X_1, \ldots, X_n) = \alpha_j$ with $\alpha_j \in \mathbb{R}$ for all $j = 1, \ldots, m$, then there is also a solution where at most m variables from each set X_i have a non-zero value.*

[16] Polynomials of this form are called set-multilinear.

Proof. This is essentially a repeated application of Carathéodory's theorem, but we will do the proof explicitly for ease of understanding by giving an iterative procedure that takes a solution to a system of m equations and eliminates a variable from a set X_i with more than m variables, i.e. sets this variable to 0. Repeatedly applying this procedure leads to the statement of this lemma.

Let $\{\xi_{i,k}\}_{i,k}$ be a solution with non-negative entries to the system of equations where more than m variables from some X_i are assigned a non-zero value by this solution. W.l.o.g. let this set be X_1. By considering the polynomials p_j as affine linear forms in $\mathbb{R}[X_2, \ldots, X_n][X_1]$ we obtain an equivalent linear system

$$P \cdot \begin{pmatrix} 1 \\ x_{1,1} \\ \vdots \\ x_{1,\kappa_1} \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix}$$

with $P \in \mathbb{R}[X_2, \ldots, X_n]^{m \times (\kappa_1 + 1)}$. Now replace the values of all variables $X_2, \ldots, X_n$ by their respective values from the solution $\{\xi_{i,k}\}_{i,k}$ to obtain the matrix P_ξ. The resulting affine linear system in the variables X_1 has κ_1 unknowns, but only $m < \kappa_1$ equations. Thus let $v \neq 0$ with $v_0 = 0$ be in the kernel of P_ξ. Additionally the system has a solution, namely $1, \xi_{1,1}, \ldots, \xi_{1,\kappa_1}$. We conclude that the entire affine subspace

$$A := \begin{pmatrix} 1 \\ \xi_{1,1} \\ \vdots \\ \xi_{1,\kappa_1} \end{pmatrix} + \lambda v$$

is a solution. Note that A cannot only contain solutions with non-negative entries and thus by continuity must contain at least one solution where all variables are non-negative and at least one variable is set to 0, for example achieved by choosing $\lambda = \max\{\frac{-\xi_{1,k}}{v_k} \mid 1 \leq k \leq \kappa_1 \text{ and } v_k \neq 0\}$.

We then iterate by removing the variables that have just been set to zero and all monomials containing them. □

With this tool in hand we first start with the rather direct application of reducing $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{DAG},\mathcal{L}_3}$ to $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{DAG},lsm,\mathcal{L}_3}$.

8.55 Lemma. $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{DAG},\mathcal{L}_3} \leq \mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{DAG},lsm,\mathcal{L}_3}$

Proof. This reduction is given by outputting the input formula φ, input graph G and $p = km + m^2$ where k is the number of arithmetic terms in φ and m is the number of exogenous variables.

Let φ be satisfied by some Markovian SCM $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U} = (U_1, \ldots, U_m), \mathbf{X})$ with graph structure G. By [ZTB22] we know that w.l.o.g. we can assume $\mathfrak{M}$ to have finite domains for all exogenous variables. Furthermore, by independence of the U_i, we know that P decomposes as

$$P(U_1{=}u_1, \ldots, U_m{=}u_m) = \prod_{i=1}^{m} P(U_i{=}u_i)$$

Now consider the system of equations

$$\sum_i \alpha_{i,j} \mathbb{P}_{\mathfrak{M}}(\delta_{i,j}) = \sum_i \sum_{\mathbf{u}:\mathfrak{M},\mathbf{u}\models\delta_{i,j}} P(\mathbf{u}) \quad \text{for each arithmetic term } \sum_i \alpha_{i,j}\mathbb{P}(\delta_{i,j}) \tag{8.56}$$

$$\sum_\ell P(U_j{=}\ell) = 1 \qquad \text{for } 1 \leq j \leq m\,. \tag{8.57}$$

This system has $k + m$ equations, each of which is set-multilinear (considering the $P(U_i{=}\ell)$ as variables after decomposing P). Thus, by Lemma 8.54 there is another non-negative solution where for each variable exogenous variable U_i at most $m + k$ values u exist with $P(U_i{=}u) > 0$. Equation (8.57), together with the non-negativity guarantees that we obtain a valid probability distribution P', while Equation (8.57) ensures that the resulting model $\mathfrak{M}' = (\mathcal{F}, P', \mathbf{U}, \mathbf{X})$ evaluates every arithmetic term identically to $\mathfrak{M}$ and thus $\mathfrak{M}' \models \varphi$. Summing over all exogenous variables, $\mathfrak{M}'$ is locally small of size at most $m(k + m) = p$. □

For $\mathrm{SAT}^{poly}_{\mathrm{DAG},\mathcal{L}_3}$ we have to be a bit more careful. We cannot directly use Lemma 8.54 because the arithmetic terms might contain multiplications and are thus no longer set-multilinear. However, using ideas similar to Lemma 8.5, without marginalization, there are only few basic terms. Applying the set-multilinear reasoning to the basic terms directly instead of the arithmetic terms then yields the local small model property.

8.58 Lemma. $\mathrm{SAT}^{poly}_{\mathrm{DAG},\mathcal{L}_3} \leq \mathrm{SAT}^{poly}_{\mathrm{DAG},lsm,\mathcal{L}_3}$

Proof. This reduction is given by outputting the input formula φ, input graph G and $p = km + m^2$ where k is the number of basic terms in φ and m is the number of exogenous variables.

Let the setup be as in Lemma 8.55, however, replace Equation (8.56) by

$$\mathbb{P}_{\mathfrak{M}}(\delta_i) = \sum_i \sum_{\mathbf{u}:\mathfrak{M},\mathbf{u}\models\delta_i} P(\mathbf{u}) \qquad \text{for each basic term } \mathbb{P}(\delta_i) \tag{8.59}$$

The resulting system has $k + m$ equations, each of which is again set-multilinear in the $P(U_i{=}\ell)$ as variables, allowing us to finish the proof identically to Lemma 8.55, with each basic term preserving its evaluation by Equation (8.59). □

Indeed, the NEXP upper bounds implied by these two reductions are tight.

8.60 Lemma. $\mathrm{SAT}^{base}_{\mathrm{MARKOV},\mathcal{L}_1}$ *is* NEXP*-hard.*

Proof. We reduce from the satisfiability of Schönfinkel-Bernays sentences over binary values with the same setup of random variables as in Lemma 8.27, with one difference: Instead of having just one random variable R_i for each k-ary relation, we add a second random variable R'_i and random variables $Z_i^1, \dots, Z_i^k$ for its k inputs (the random variables R_i^j for the occurrences of the relation are not impacted by this and still exist as usual). We use them to make the relations deterministic.

We start by modeling the existential and universal quantifications of $\mathbf{x}$ and $\mathbf{y}$ respectively. This is achieved by ensuring none of the X_i or Y_i have any parents in our graph – resulting in their independence – and the following constraints:

$$\mathbb{P}(X_i{=}0) = 1 \vee \mathbb{P}(X_i{=}1) = 1 \qquad \text{for all } i = 1, \dots, n \tag{8.61}$$

$$\mathbb{P}(Y_i{=}0) = \frac{1}{2} \wedge \mathbb{P}(Y_i{=}1) = \frac{1}{2} \qquad \text{for all } i = 1, \dots, n \tag{8.62}$$

As a result, the values of the X_i are deterministic, while all combinations of values for the Y_i have equal probability.

To deal with the relations we first add the constraints[17]

$$\mathbb{P}(R_i{\neq}R'_i) = 0 \tag{8.63}$$

and

$$\mathbb{P}(Z_i^\ell{=}0) = \mathbb{P}(Z_i^\ell{=}1) = \frac{1}{2} \qquad \text{for all } \ell = 1, \dots, k \tag{8.64}$$

for each relation R_i. Furthermore, in our graph we make $Z_i^1, \dots, Z_i^k$ the only parents of R_i and R'_i. This enforces that the value of R_i depends deterministically only on its parents, i.e. its inputs.

[17] This first set of constraints (Equation (8.63)) and the random variables R'_i are not technically required, however they simplify the correctness argument a bit by ensuring full determinism of the relations for all inputs and not just for the inputs that are used in a solution.

As a next step, we ensure that R_i and each of the R_i^j behave consistently, i.e. produce the same output for the same inputs. We achieve this by adding the following constraint for each occurrence R_i^j. For simplicity we denote the list of inputs of R_i^j as $\mathbf{T}_i^j$ and the list of inputs of R_i as $\mathbf{Z}_i$.

$$\mathbb{P}(\mathbf{Z}_i{=}\mathbf{T}_i^j \wedge R_i{\neq}R_i^j) = 0 \tag{8.65}$$

Finally, let ψ' be obtained from ψ by replacing equality and relations on the Boolean values with the corresponding definitions for the random variables:

$$\mathbb{P}(\psi') = 1 \tag{8.66}$$

Let $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ be a true sentence. We construct an SCM $\mathfrak{M}$ with the graph structure described above, satisfying all the equations. First, the values for the $\mathbf{X}$ are deterministic

according to the choice of $\mathbf{x}$ that satisfies the formula, while the $\mathbf{Y}$ are uniformly distributed. This way Equations (8.61) and (8.62) are satisfied. All R_i, R'_i and R_i^j variables are deterministic (i.e. ignoring their own exogenous variable), depending only on the values of their parameters in the way specified by the respective relation. Since R_i and R'_i share the same inputs, Equation (8.63) is satisfied. Similarly, whenever some R_i and R_i^j have the same inputs, their outputs agree, which satisfies Equation (8.65). Further, by choosing all Z_i^ℓ as independent uniform variables, Equation (8.64) is satisfied. Lastly, no matter the choice of $\mathbf{y}$, ψ is valid and thus ψ' is valid for any choice of the values for $\mathbf{Y}$, satisfying Equation (8.66).

For the other direction, let $\mathfrak{M}$ be an SCM satisfying the constructed equations and graph structure. Clearly Equation (8.61) ensures that the $\mathbf{X}$ have a fixed value. By Equation (8.66), no matter the value of the $\mathbf{Y}$ (all combinations thereof exist by Equation (8.62)), ψ' is valid. If we can now show that the relations are consistently represented in ψ', we are in the same situation as above and can reverse the process to obtain $\mathbf{x}$ and the relations that satisfy our Schönfinkel-Bernays sentence. Firstly, both R_i and R'_i share all their parents, except for their exogenous variables U_{R_i} and $U_{R'_i}$. Since U_{R_i} and $U_{R'_i}$ are independent of each other, Equation (8.63) forces R_i and R'_i to be independent of the values of U_{R_i} and $U_{R'_i}$, i.e. deterministically determined by their parents. Second, if there were two variables R_i^j and $R_i^{j'}$ with different values, even though their inputs are equal ($\mathbf{T}_i^j = \mathbf{T}_i^{j'}$), then, for $\mathbf{Z}_i = \mathbf{T}_i^j = \mathbf{T}_i^{j'}$, Equation (8.65) is violated. Note that this requires the probability for $\mathbf{Z}_i = \mathbf{T}_i^j$ to be positive, which is achieved by Equation (8.64) combined with the fact that the $\mathbf{Z}_i$ exclusively depend on their own independent exogenous variables. □

Combining Lemmas 8.55, 8.58 and 8.60 we get the first set of completeness results:

8.67 Theorem. $\mathrm{SAT}^*_{\mathrm{DAG},\mathcal{L}_i}$ *and* $\mathrm{SAT}^*_{\mathrm{MARKOV},\mathcal{L}_i}$ *are* NEXP*-complete for* $i = 1, 2, 3$ *and* $* \in \{\mathit{base}, \mathit{base}\langle\Sigma\rangle, \mathit{lin}, \mathit{lin}\langle\Sigma\rangle, \mathit{poly}\}$.

It is perhaps a bit surprising that we have the same complexity for all of these. This is the only case where we observe the exponential sums to not increase the complexity of the linear languages.

Once we go on further to the languages $\mathrm{SAT}^{\mathit{poly}\langle\Sigma\rangle}_{\mathrm{DAG},\mathcal{L}_i}$/$\mathrm{SAT}^{\mathit{poly}\langle\Sigma\rangle}_{\mathrm{MARKOV},\mathcal{L}_i}$, we no longer have either of the small model properties: there are potentially exponentially many different probabilistic terms and the arithmetic terms are no longer set-multilinear in the exogenous probabilities. Indeed, these languages no longer have the local small model property and thus their complexity jumps up to $\mathsf{NEXP}_{\mathrm{real}}$.

We start with the observation that $\mathrm{SAT}^*_{\mathcal{L}_1}$ is not harder than $\mathrm{SAT}^*_{\mathrm{MARKOV},\mathcal{L}_1}$ for any $* \in \mathit{Lab}$ and thus in particular all of $\mathrm{SAT}^{\mathit{poly}\langle\Sigma\rangle}_{\mathrm{MARKOV},\mathcal{L}_i}$ inherit the $\mathsf{NEXP}_{\mathrm{real}}$-hardness of $\mathrm{SAT}^{\mathit{poly}\langle\Sigma\rangle}_{\mathcal{L}_1}$.

8.68 Proposition. *For all $* \in Lab$ and for any $\varphi \in \mathcal{L}_1^*$ over variables $X_1, \dots, X_n$, it is true that $\varphi \in \mathrm{SAT}^*_{\mathcal{L}_1}$ if and only if $(\varphi, \mathcal{G}_n) \in \mathrm{SAT}^*_{\textsc{Markov},\mathcal{L}_1}$, where $\mathcal{G}_n$ denotes a complete DAG over nodes $X_1, \dots, X_n$, with edges $X_i \to X_j$ for all $1 \leq i < j \leq n$.*

Proof. Assume that φ is satisfiable by an arbitrary SCM $\mathfrak{M}$. Using the chain rule of probability we can factor the joint probability distribution over the endogenous variables $X_1, \dots, X_n$ as

$$P(x_1, \dots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_1, \dots, x_{i-1}) \,.$$

In particular this directly gives rise to an SCM $\mathfrak{M}'$ with identical joint probability distribution over the endogenous variables where every X_i only depends on $X_1, \dots, X_{i-1}$ and its own independent exogenous random variable U_i. Since $\varphi \in \mathcal{L}_1^*$, it will thus evaluate identically on $\mathfrak{M}$ and $\mathfrak{M}'$.

The reverse implication is obvious. □

Note that this proof in essence just converts the probability distribution over the endogenous variables to a Bayesian network, a different model that represents Markovian SCMs over some DAG G as conditional probability tables. They, however, are weaker in the sense that they often can't answer interventional queries in the absence of specific criteria, for example the back-door criterion [Pea09].

For an upper bound, we observe that any of our formulas reference at most exponentially many basic terms, allowing us to restrict ourselves to SCMs $\mathfrak{M}$ only consisting of exponentially many values: Discrete values for the functional dependencies and real values for the joint probability distribution of the exogenous variables. This allows a $\mathsf{NEXP}_{\mathrm{real}}$ machine to simply guess the entire model $\mathfrak{M}$ and check $\mathfrak{M} \models \varphi$.

8.69 Theorem. $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathrm{DAG},\mathcal{L}_i}$ *and* $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\textsc{Markov},\mathcal{L}_i}$ *are* NEXP_{real}*-complete for any* $i = 1, 2, 3$.

Proof. $\mathsf{NEXP}_{\mathrm{real}}$-hardness follows from Proposition 8.68. We continue with containment in $\mathsf{NEXP}_{\mathrm{real}}$: By [ZTB22, Theorem 1], for any SCM $\mathfrak{M}$ we can, w.l.o.g. assume the size of the domain of each exogenous variable to be bounded by c^{c^n}. Using the same argument as in Lemma 8.58 we can strengthen this bound using Lemma 8.54: After expanding all exponential sums, there are only single-exponentially (in c and n) many different basic terms, each set-multilinear in the probabilities of the exogenous variables. Thus we can assume, by Lemma 8.54, each exogenous variable to have a single-exponentially sized domain.

A $\mathsf{NEXP}_{\mathrm{real}}$ machine can now simply guess the probability distribution of each exogenous variable (as real values), guess the functional dependencies representing the input DAG (as discrete values) and explicitly verify $\mathfrak{M} \models \varphi$ after expanding all exponential sums. □

8.6.2 Semi-Markovian Models

If, instead of Markovian models, we consider just semi-Markovian models with required graph structures, we get a different picture. As a reminder, the difference between Markovian and semi-Markovian here is that the functions $\mathcal{F}$ can depend on all exogenous random variables for semi-Markovian models. In particular we can, by Lemma 8.3, assume that semi-Markovian models only have a single exogenous random variable.

As a first observation, the versions of our satisfiability problems with constrained semi-Markovian graph structures are always at least as hard as their unconstrained variants. All reductions in Sections 8.3 to 8.5 either don't care about the ordering of the exogenous variables or explicitly enforce a specific ordering. In either case, the dependencies can be chosen as a complete graph, compatible with the required ordering if needed.

Additionally, the upper bounds in Section 8.3 (as can be best seen applying the ideas from Lemma 8.12) for the languages without marginalization can check at every step whether the guesses of the model violate the graph structure. Combined with Lemmas 8.70 and 8.71 we will see that this is not an isolated phenomenon, but we always have $\mathrm{SAT}^{*}_{\mathcal{L}_i} \equiv_P \mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_i}$ for $i = 1, 3$ and any $* \in \mathit{Lab}$, leaving the interventional case as the only open question. And indeed, this is where the only difference in complexity appears: $\mathrm{SAT}^{\mathit{base}\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ and $\mathrm{SAT}^{\mathit{lin}\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ increase from the PSPACE-completeness of their unconstrained variants all the way to NEXP-completeness.

We start proving the marginalization cases by first observing that purely probabilistic languages are not powerful enough to distinguish between different underlying graph structures:

8.70 Lemma. $\mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_1} \equiv_P \mathrm{SAT}^{*}_{\mathcal{L}_1}$ *for any* $* \in \mathit{Lab}$.

Proof. Let $\mathfrak{M} = (\mathcal{F}, P, \mathbf{U}, \mathbf{X})$ be any semi-Markovian SCM. W.l.o.g. we can assume $\mathbf{U} = \{U\}$ by Lemma 8.3 and $X_1 \prec \ldots \prec X_n$. We now construct a probabilistically equivalent SCM $\mathfrak{M}' = (\mathcal{F}', P, \mathbf{U}, \mathbf{X})$ that agrees with every graph structure. We recursively express each of the F_i only as a dependence on the exogenous variable. Start with $F'_1 := F_1$. Then $F'_i(u) := F_i(F'_1(u), \ldots, F'_{i-1}(u), u)$, ignoring F'_j corresponding to non-parents of X_i in $\mathfrak{M}$. Now clearly $\mathfrak{M}$ and $\mathfrak{M}'$ induce the same joint probability distribution over the endogenous variables and thus both satisfy the same purely probabilistic formulas. □

Counterfactual languages on the other hand are strong enough to be able to enforce an explicit graph structure directly in the formula using the marginalization. That is, for each variable X_i, the formula can encode which variables are the parents of X_i by requiring that, for fixed values of the exogenous variables, X_i remains constant if the parents remain constant. This condition is checked by comparing an intervention on just the parents against an intervention on all variables. A marginalization can repeat this for all possible interventions.

8.71 Lemma. $\mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_3} \leq_P \mathrm{SAT}^{*}_{\mathcal{L}_3}$ *for any* $* \in \{base\langle\Sigma\rangle, lin\langle\Sigma\rangle, poly\langle\Sigma\rangle\}$.

Proof. We encode the structure of the graph $\mathcal{G}$ directly into the formula.

Let X_i be an endogenous variable and let $\mathbf{Pa}_i$ be all the endogenous variables that are direct predecessors of X_i in $\mathcal{G}$. We then add the constraint

$$\sum_{\mathbf{v}} \mathbb{P}([\mathbf{pa}_i]X_i \neq [\mathbf{v} \setminus x_i]X_i) = 0$$

to ensure that X_i only depends on $\mathbf{Pa}_i$. The sum iterates over all possible values $\mathbf{v}$ of the endogenous variables. □

As a consequence of the discussion in this section thus far, combined with the trivial inclusions $\mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_1} \leq_P \mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_2} \leq_P \mathrm{SAT}^{*}_{\mathrm{sMarkov},\mathcal{L}_3}$ for any $* \in Lab$, we have

8.72 Corollary. *For* $i = 1, 2, 3$,

- $\mathrm{SAT}^{base}_{\mathrm{sMarkov},\mathcal{L}_i}$ *and* $\mathrm{SAT}^{lin}_{\mathrm{sMarkov},\mathcal{L}_i}$ *are* NP*-complete,*
- $\mathrm{SAT}^{poly}_{\mathrm{sMarkov},\mathcal{L}_i}$ *is* NP_{real}*-complete, and*
- $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_i}$ *is* NEXP_{real}*-complete.*

Furthermore

- $\mathrm{SAT}^{base}_{\mathrm{sMarkov},\mathcal{L}_1}$ *and* $\mathrm{SAT}^{lin}_{\mathrm{sMarkov},\mathcal{L}_1}$ *are* $\mathsf{NP}^{\mathsf{PP}}$*-complete, and*
- $\mathrm{SAT}^{base}_{\mathrm{sMarkov},\mathcal{L}_3}$ *and* $\mathrm{SAT}^{lin}_{\mathrm{sMarkov},\mathcal{L}_3}$ *are* NEXP*-complete.*

Remains to analyze the complexity of $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$. They are neither weak enough to be independent of the graph structure nor strong enough to be able to distinguish graph structures exactly, which leads to a complexity jump to NEXP compared to the PSPACE-complexity of $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathcal{L}_2}$ and $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathcal{L}_2}$.

8.73 Theorem. $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ *and* $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{sMarkov},\mathcal{L}_2}$ *are* NEXP*-complete.*

Proof. We again reduce from the satisfiability of Schönfinkel-Bernays sentences, reusing the same random variable setup from Lemma 8.27. Additionally, for each k-ary relation $R_i(z_1, \ldots, z_k)$, we add random variables $Z_i^1, \ldots, Z_i^k$ for the inputs.

We use the graph G to ensure that R_i only depends on its arguments. For this, the incoming edges to R_i are exactly the edges from $Z_i^1, \ldots, Z_i^k$. In the same way we ensure that R_i^j only depends on its arguments by making the only incoming edges to R_i^j exactly the edges from $T_1, \ldots, T_k$ (the inputs used for the j-th occurrence of the i-th relation).

We add the following constraint to ensure that R_i^j and R_i have an equal value for equal arguments:

$$\sum_{t_1,\ldots,t_k} \mathbb{P}([T_1{=}t_1,\ldots,T_k{=}t_k, Z_i^1{=}t_1,\ldots,Z_i^k{=}t_k](R_i^j{\neq}R_i)) = 0. \tag{8.74}$$

We continue by ensuring that the values of $\mathbf{X}$ are not affected by the values of $\mathbf{Y}$ by making the $\mathbf{X}$ have no predecessors in G.

Let ψ' be obtained from ψ by replacing equality and relations on the Boolean values with the corresponding definitions for the random variables:

$$\sum_{\mathbf{y}} \mathbb{P}([\mathbf{y}]\psi') = 2^n \tag{8.75}$$

Suppose the $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\mathrm{sMARKOV},\mathcal{L}_2}$ instance is satisfied by some model $\mathfrak{M}$. We need to show that $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ is satisfiable. First note that the values $\mathbf{x}$ of the variables $\mathbf{X}$ deterministically depend only on the values $\mathbf{u}$ of the exogenous variables. Choose any value $\mathbf{u}$ (and thus corresponding $\mathbf{x}$) that contributes a positive probability to any of the $\mathbb{P}([\mathbf{y}]\psi')$ in Equation (8.75). If there was any $\mathbf{x}$ chosen this way that would not satisfy ψ for all values of $\mathbf{y}$, $\mathbb{P}([\mathbf{y}]\psi')$ would be less than 1 for these values of $\mathbf{x}$ (determined by $\mathbf{u}$) and $\mathbf{y}$, and Equation (8.75) would not be satisfied. The relations R_i are chosen as the values of the random variables R_i. Equation (8.74) now enforces two things, together with the graph structure. First of all, the R_i^j and R_i depend deterministically only on their parameters and possibly the $\mathbf{u}$ values. Since we have fixed the values $\mathbf{u}$, they only depend on their parameters for the rest of this proof. Secondly, all occurrences of R_i are given the same value. Thus, $\mathbb{P}([\mathbf{y}]\psi') = 1$ implies ψ that $\mathbf{x}$ and $\mathbf{y}$ satisfy ψ and we conclude that $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ is a true sentence.

Suppose $\exists\mathbf{x}\,\forall\mathbf{y}\,\psi$ is satisfiable. Create a deterministic model $\mathfrak{M}$ by choosing the values of the random variables $\mathbf{X}$ to the values chosen by $\exists\mathbf{x}$. The random variables R_i (and R_i^j) are chosen as functions depending on their parameters, behaving exactly like the relation R_i. These are allowed dependencies according to our constructed graph structure. Clearly Equation (8.74) is satisfied by this. The remaining random variables (the $\mathbf{y}$) can be chosen arbitrarily as constants. Their values do not matter since they are overridden by interventions whenever their value would become relevant. It remains to consider Equation (8.75), but this one is satisfied because ψ is satisfied for all values of $\mathbf{y}$.

Finally, $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{sMARKOV},\mathcal{L}_2}$ is a special case of $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\mathrm{sMARKOV},\mathcal{L}_3}$ and thus can be solved in NEXP by Corollary 8.72. □

Small Models

In contrast to Markovian models, for semi-Markovian models we were able to use Lemma 8.3 to assume there is always just a single exogenous variable. Due to this we investigate the complexity of the combination of semi-Markovian models and small models here.

We can use the exact same arguments as in Section 8.6.2 to see that the complexity for $\mathrm{SAT}^{*}_{\text{sMarkov},sm,\mathcal{L}_i}$ for $i = 1, 3$ remains precisely the same as the variants with a small model restriction, but no restriction on the causal diagram. Furthermore, the reductions in Theorems 8.47 and 8.73 both build a deterministic, and thus small, model. This immediately gives us the full classification, also seen in Table 8.11

8.76 Corollary. *For $i = 1, 2, 3$,*

- $\mathrm{SAT}^{base}_{\text{sMarkov},sm,\mathcal{L}_i}$ *and* $\mathrm{SAT}^{lin}_{\text{sMarkov},sm,\mathcal{L}_i}$ *are* NP*-complete, and*
- $\mathrm{SAT}^{poly}_{\text{sMarkov},sm,\mathcal{L}_i}$ *is* NP_{real}*-complete.*

Furthermore, for $i = 2, 3$,

- $\mathrm{SAT}^{base}_{\text{sMarkov},sm,\mathcal{L}_i}$ *and* $\mathrm{SAT}^{lin}_{\text{sMarkov},sm,\mathcal{L}_i}$ *are* NEXP*-complete*
- $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\text{sMarkov},sm,\mathcal{L}_i}$ *is* NEXP*-complete.*

Finally

- $\mathrm{SAT}^{base\langle\Sigma\rangle}_{\text{sMarkov},sm,\mathcal{L}_1}$ *and* $\mathrm{SAT}^{lin\langle\Sigma\rangle}_{\text{sMarkov},sm,\mathcal{L}_1}$ *are* $\mathsf{NP}^{\mathsf{PP}}$*-complete, and*
- $\mathrm{SAT}^{poly\langle\Sigma\rangle}_{\text{sMarkov},sm,\mathcal{L}_1}$ *is* $\mathsf{NP}^{\mathsf{VNP}_{\mathbb{R}}}_{real}$*-complete.*

8.7 Summary

For the readers convenience we give a summary over the complexity of all variations, grouped by their constraints on the model. See

- Table 8.6 for fully unconstrained models ($\mathrm{SAT}^{*}_{\mathcal{L}_i}$),
- Table 8.7 for small models ($\mathrm{SAT}^{*}_{sm,\mathcal{L}_i}$),
- Table 8.8 for explicit arbitrary causal diagrams ($\mathrm{SAT}^{*}_{\text{DAG},\mathcal{L}_i}$),
- Table 8.9 for explicit Markovian causal diagrams ($\mathrm{SAT}^{*}_{\text{Markov},\mathcal{L}_i}$),
- Table 8.10 for explicit semi-Markovian causal diagrams ($\mathrm{SAT}^{*}_{\text{sMarkov},\mathcal{L}_i}$), and
- Table 8.11 for explicit semi-Markovian causal diagrams combined with small models ($\mathrm{SAT}^{*}_{\text{sMarkov},sm,\mathcal{L}_i}$).

Terms	Probabilistic	Interventional	Counterfactual
basic	NP		
lin			
poly	$\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$		
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$	PSPACE	NEXP
lin & marg.			
poly & marg.	$\mathsf{succ}\text{-}\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$		

Table 8.6: Completeness results for the satisfiability problems with a fully unconstrained model ($\text{SAT}^*_{\mathcal{L}_i}$). Summarizes the results of Sections 8.3 to 8.5.

Terms	Probabilistic	Interventional	Counterfactual
basic	NP		
lin			
poly	$\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$		
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$	PSPACE	NEXP
lin & marg.			
poly & marg.	$\exists\mathbb{R}^{\Sigma} = \mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$	NEXP	

Table 8.7: Completeness results for the satisfiability problems with a small model ($\text{SAT}^*_{sm,\mathcal{L}_i}$). Summarizes the results of Sections 8.3.1, 8.4 and 8.5.2.

Terms	Probabilistic	Interventional	Counterfactual
basic	NEXP		
lin			
poly			
basic & marg.	NEXP		
lin & marg.			
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$		

Table 8.8: Completeness results for the satisfiability problems with given full graph structure for the model ($\text{SAT}^{*}_{\text{DAG},\mathcal{L}_i}$). Summarizes the results of Section 8.6.1.

Terms	Probabilistic	Interventional	Counterfactual
basic	NEXP		
lin			
poly			
basic & marg.	NEXP		
lin & marg.			
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$		

Table 8.9: Completeness results for the satisfiability problems with given Markovian graph structure for the model ($\text{SAT}^{*}_{\text{MARKOV},\mathcal{L}_i}$). Summarizes the results of Section 8.6.1.

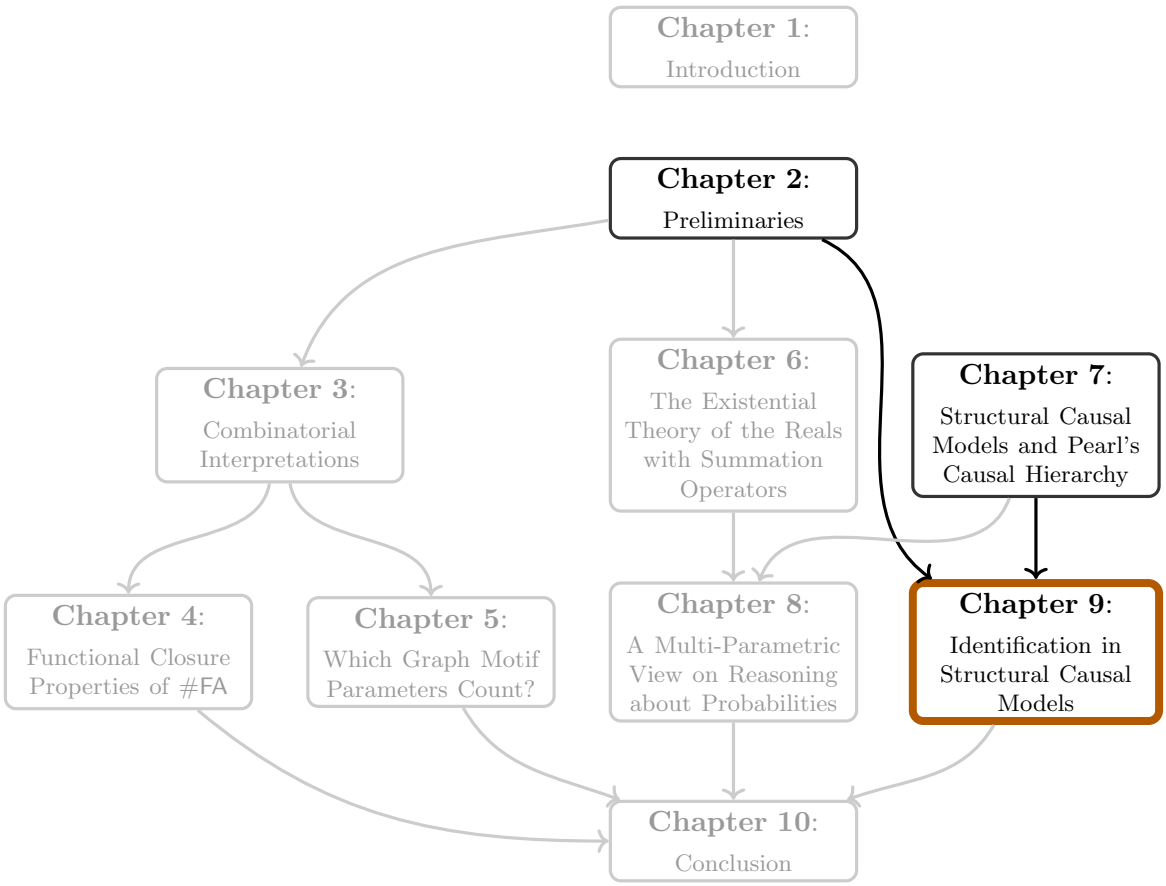
Terms	Probabilistic	Interventional	Counterfactual
basic	NP		
lin			
poly	$\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$		
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$	NEXP	
lin & marg.			
poly & marg.	succ-$\exists\mathbb{R} = \mathsf{NEXP}_{\text{real}}$		

Table 8.10: Completeness results for the satisfiability problems with given semi-Markovian graph structure for the model ($\text{SAT}^*_{\text{sMARKOV},\mathcal{L}_i}$). Summarizes the results of Section 8.6.2.

Terms	Probabilistic	Interventional	Counterfactual
basic	NP		
lin			
poly	$\exists\mathbb{R} = \mathsf{NP}_{\text{real}}$		
basic & marg.	$\mathsf{NP}^{\mathsf{PP}}$	NEXP	
lin & marg.			
poly & marg.	$\exists\mathbb{R}^{\Sigma} = \mathsf{NP}^{\mathsf{VNP}_{\mathbb{R}}}_{\text{real}}$	NEXP	

Table 8.11: Completeness results for the satisfiability problems with given semi-Markovian graph structure for the model, combined with a small model ($\text{SAT}^*_{\text{sMARKOV},sm,\mathcal{L}_i}$). Summarizes the results of Section 8.6.2.

We leave open the precise complexity of combining an explicit arbitrary or Markovian causal diagram with a small model, however, as discussed at the start of Section 8.6.1, this enforces very few (non-trivial) exogenous variables (and thus very few endogenous variables in the Markovian case). A further open question is the complexity of more of the locally small model variants, introduced in Section 8.6.1.



Identification in Structural Causal Models

9.1 Introduction

Recognizing and predicting the causal effects and distinguishing them from purely statistical correlations is an important task of empirical sciences. For example, identifying the causes of disease and health outcomes is of great significance in developing new disease prevention and treatment strategies. A common approach for establishing causal effects is through randomized controlled trials [Fis36] – called the gold standard of experimentation – which, however, requires physical intervention in the examined system. Therefore, in many practical applications, experimentation is not always possible due to cost, ethical constraints, or technical feasibility. E.g., to learn the effects of radiation on human health one cannot conduct interventional studies involving human participants. In such cases, a researcher may use an alternative approach and establish cause-effect relationships by combining existing observed data with the knowledge of the structure of the system under study. This is called the problem of *identification* in causal inference [Pea09] and the approach is commonly used in various fields, including modern ML.

A key ingredient of this framework is the way the underlying structure models the true mechanism behind the system. In general, this is done using structural causal models (SCMs) [Pea09; BCII22], as introduced in Chapter 7. In this chapter, we focus on the problem of identification in linear SCMs, known also as structural equation models (SEMs) [Bol89; Dun75]. They represent the causal relationships between observed random variables assuming that each variable X_i, with $i = 1, \ldots, n$ is linearly dependent on the remaining variables and an unobserved error term ε_i of normal distribution with zero mean: $X_j = \sum_i \lambda_{i,j} X_i + \varepsilon_j$. The model implies the existence of some covariance matrix $\Omega = (\omega_{i,j})$ between the error terms. In this paper, we consider *recursive models*, i.e., we assume that, for all $i > j$, we have $\lambda_{i,j} = 0$.

Linear SCMs can be represented as a graph with nodes $\{1, \ldots, n\}$ corresponding to variables $X_1, \ldots, X_n$ and with directed and bidirected edges. A directed edge $i \to j$ represents a linear influence $\lambda_{i,j} \neq 0$ of a parent node i on its child j. A bidirected edge $i \leftrightarrow j$ represents a correlation $\omega_{i,j} \neq 0$ between error terms ε_i and ε_j (cf. Figure 9.1).

Writing the coefficients of all directed edges as an adjacency matrix $\Lambda = (\lambda_{i,j})$ and the coefficients of all bidirected edges as an adjacency matrix $\Omega = (\omega_{i,j})$, the covariances $\sigma_{i,j}$ between each pair of observed variables X_i and X_j can be calculated as the matrix $\Sigma = (\sigma_{i,j})$:

$$\Sigma = (I - \Lambda)^{-T} \Omega (I - \Lambda)^{-1}, \tag{9.1}$$

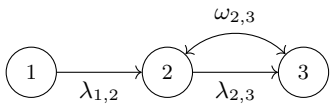


Figure 9.1: An IV example.

where I is the identity matrix [FDD12]. Knowledge of the parameters $\lambda_{i,j}$ allows for the prediction of all causal effects in the system. The key task here is to learn Λ from the observed covariances Σ assuming Ω remains unknown. This leads to the formulation of the identification problem in linear SCMs as solving for the parameter Λ using equation (9.1). If the problem asks to find symbolic solutions involving only symbols $\sigma_{i,j}$, we call it the problem of *symbolic identification*. In the case when the parameter can be determined uniquely almost everywhere using Σ alone, we call the instance to be *generically identifiable* (for a formal definition, see Section 9.2). If the goal is to find, for a given Σ of rational numbers, the numerical solutions, we call it the problem of *numerical identification*. In this chapter, we study the computational complexity of both generic and numerical identification.

Previous Work. The identification in linear SCMs and its applications have been a subject of research interest for many decades, including the early work in econometrics and agricultural sciences [Wri21; Wri28; Fis66; BT84]. Currently, it seems, that one of the most challenging tasks in this field is providing efficient computational methods to find solutions, both for symbolic and for numeric variants, or providing evidence that the problems are computationally intractable.

The generic identification can be computed using standard algebraic tools for solving symbolic polynomial equations (9.1). Such an approach provides a *sound* and *complete* method, i.e., it is guaranteed to identify all identifiable instances. However, common algorithms for solving such equations usually use Gröbner basis computation, whose time complexity is doubly exponential in the worst case [GSS10]. So far, it has remained widely open whether the double exponential function is a sharp upper bound on the computational complexity of the generic identifiability.

Most approaches to solving the problem in practice are based on instrumental variables, in which the causal direct effect is identified as a fraction of two covariances [Wri28; BT84]. For example, in the linear model shown in Figure 9.1, one can calculate first $\lambda_{1,2} = \sigma_{1,2}$ and then $\lambda_{2,3} = \frac{\lambda_{1,2}\lambda_{2,3}}{\lambda_{1,2}} = \frac{\sigma_{1,3}}{\sigma_{1,2}}$. The variable X_1 is then called an instrumental variable (IV). This method is sound but not complete, that is, when it identifies a parameter, then it is always correct. However, when the method fails due to a missing IV, then the parameter might still be identified by other means. This approach has inspired intensive research aimed at providing computational methods that may not be complete but enable efficient algorithms and identify a significantly large number of cases.

Conditional IVs (cIVs) are one of the most natural extensions of simple IVs [BT84; Pea01]. The corresponding identification method is based on an efficient, polynomial time algorithm for finding conditional IVs [ZTL15]. More complex criteria and methods proposed in the literature, which are also accompanied by polynomial time algorithms, involve instrumental sets (IS) [BP02b] half-treks (HTC) [FDD12], instrumental cutsets (ICs) [KCB19], auxiliary instrumental variables (aIVs) [CPB15]. The generalized HTC

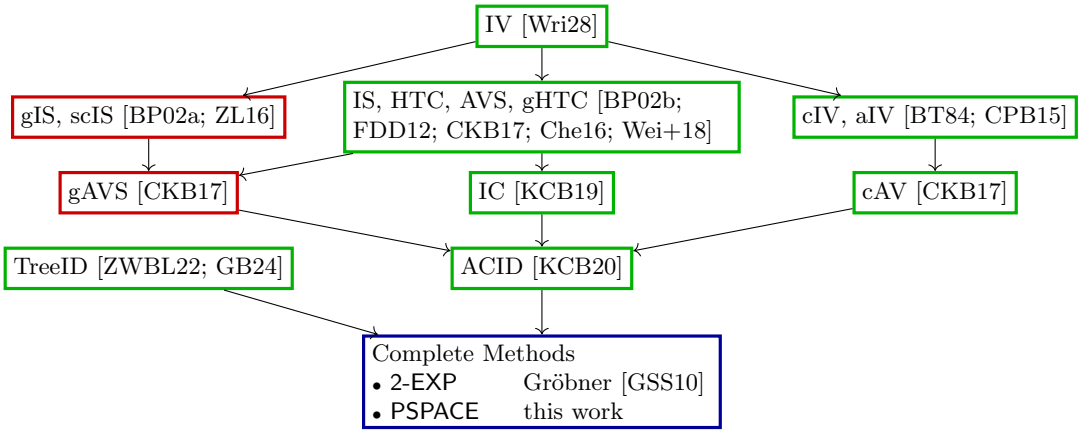


Figure 9.2: Methods for generic identification in linear SCMs. An arrow from methods $A \to B$ means that B subsume all methods A, i.e., any instance that can be identified by any of methods A can be identified by method B and this inclusion is proper. Green boxes mean there exist polynomial-time algorithms to apply the method, a red box means no such algorithm is known or the method has been proven to be NP-hard. The blue box includes the complete methods.

(gHTC) [Che16; Wei+18] and auxiliary variables (AVS) [Che16; CKB17] can be implemented in polynomial time provided that the number of incoming edges to each node in the causal graph is bounded. The methods based on generalized IS (gIS), a simplified version of the criterion (scIS), and generalized AVS (gAVS) appeared to be computationally intractable [BP02b; Bri10; BP02a; ZL16; CKB17]. The auxiliary cutsets (ACID) algorithm [KCB20] subsumes all the above methods in the sense that it covers all the instances identified by them. Recently [ZWBL22; GB24] provide the TreeID algorithm for identification in tree-shaped linear models. TreeID is incomparable since it is complete for the subclass of tree-like SCMs. However, TreeID does not work for general SCMs, which is the focus of our work. Figure 9.2 summarizes these results.

Numerical parameter identification, known in the literature as the estimation of the parameters of structural equation models, has been the subject of a considerable amount of research, which has resulted in significant progress in theoretical understanding and development of estimation methods [AR56; Jör69; Bro74; Mut83; Bol89; HBK92]. Currently, in practical applications (e.g. in econometrics, psychometrics, or biometrics), methods based on maximum likelihood (ML) or generalized least squares estimator (GLS) are commonly used to find model parameters. However, despite the great importance of this problem and considerable effort in method development, the computational complexity of the parameter estimation problem remains unexplored. In our work, we provide, to our knowledge, the first hardness result for a very basic variant of the SEM parameter estimation problem, which we call numerical identification.

Our Contribution. We improve significantly the best-known upper bound on the computational complexity for generic identification and provide evidence that parameter identification is computationally hard in general. In more detail, our contributions are as follows:

- We provide a polynomial-space algorithm for sound and complete generic parameter identification in linear models. This gives an exponential running time which vastly improves the state-of-the-art double exponential time method using Gröbner basis. In our approach, we formulate generic identifiability as a formula over real variables with universal quantifiers and then use Renegar's algorithm [Ren92].
- Our constructive technique allows us to prove an $\forall\mathbb{R}$ upper bound for generic identifiability.
- We prove that numerical identification is hard for the complexity class $\forall\mathbb{R}$. In particular, the problem is co-NP-hard. Our complexity characterization is quite precise since we show a (promise) $\forall\mathbb{R}$ upper bound for numerical identification. To the best of our knowledge, this is the first hardness result for some notion of identifiability.

  This implies that numerical identifiability can be decided in polynomial space.

9.2 Preliminaries

9.2.1 The Problems of Identification in Linear Causal Models

A mixed graph is a triple $G = (V, D, B)$ where $V := \{1, \ldots, n\}$ is a finite set of nodes and $D \subseteq V \times V$ and $B \subseteq \binom{V}{2}$ are two sets of directed and bidirected edges, respectively. Let $\mathbb{R}^D$ be the set of matrices $\Lambda = (\lambda_{i,j}) \in \mathbb{R}^{n\times n}$ with $\lambda_{i,j} = 0$ if $i \to j$ is not in D and let $\mathrm{PD}(n)$ denote the cone of positive definite $n \times n$ matrices. Let $\mathrm{PD}(B)$ be the set of matrices $\Omega = (\omega_{i,j}) \in \mathrm{PD}(n)$ with $\omega_{i,j} = 0$ if $i \neq j$ and $i \leftrightarrow j$ is not an edge in B. We will only consider recursive models, i.e. we assume that, for all $i > j$, we have $\lambda_{i,j} = 0$. Thus, the directed graph (V, D) accompanied with the model is acyclic. We will assume w.l.o.g. that the nodes are topologically sorted, i.e., there are no edges $i \to j$ with $i > j$.

Denote by $\mathcal{N}_n(\mu, \Sigma)$ the multivariate normal distribution with mean $\mu \in \mathbb{R}^n$ and covariance matrix Σ. The linear SCMs $\mathcal{M}(G)$ associated with $G = (V, D, B)$ is the family of multivariate normal distributions $\mathcal{N}_n(0, \Sigma)$ with Σ satisfying equation (9.1), for $\Lambda \in \mathbb{R}^D$ and $\Omega \in \mathrm{PD}(B)$. A model in $\mathcal{M}(G)$ is specified in a natural way in terms of a system of linear structural equations: $X_j = \sum_{i\in \mathrm{pa}(j)} \lambda_{i,j} X_i + \varepsilon_j$, for $j = 1, \ldots, n$, where $\mathrm{pa}(j)$ denotes the parents of j in G. If $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$ is a random vector with the multivariate normal distribution $\mathcal{N}_n(0, \Sigma)$ and $\Lambda \in \mathbb{R}^D$, then the random vector $X = (X_1, \ldots, X_n)$ is well defined as a solution to the equation system and follows a centered multivariate normal distribution with covariance matrix $(I - \Lambda)^{-T}\Omega(I - \Lambda)^{-1}$ (see, e.g. [DFS11]).

For a given (acyclic) mixed graph $G = (V, D, B)$, define the parametrization map

$$\phi_G : (\Lambda, \Omega) \mapsto (I - \Lambda)^{-T} \Omega (I - \Lambda)^{-1}$$

and let $\Theta := \mathbb{R}^D \times \mathrm{PD}(B)$. We say that G is *globally* identifiable if ϕ_G is injective on Θ [DFS11].

Global identification can be decided easily, see [DFS11, Theorem 2]. However, it is a very strong property. For instance, as seen in the introduction, in Figure 9.1, we can recover the parameter $\lambda_{2,3}$ as $\frac{\sigma_{1,3}}{\sigma_{1,2}}$. If $\sigma_{1,2} = 0$, then the identification fails, so the instance is not globally identifiable. But identification fails only in the (very unlikely) case that $\sigma_{1,2} = 0$. This leads to the concept of *generic identifiability*:

9.2 Definition (Generic Identifiability, [FDD12])**.** The mixed graph G is said to be *generically* identifiable if ϕ_G is injective on the complement $\Theta \setminus \mathcal{V}$ of a proper (i.e., strict) algebraic subset $\mathcal{V} \subsetneq \Theta$.

Any proper algebraic subset $\mathcal{V} \subsetneq \Theta$ has a lower dimension than Θ and is consequently of measure zero.

Given matrices $\Lambda_0 \in \mathbb{R}^D$ and $\Omega_0 \in \mathrm{PD}(B)$, the corresponding *fiber* is defined by

$$\mathcal{F}_G(\Lambda_0, \Omega_0) = \{(\Lambda, \Omega) \mid \phi_G(\Lambda, \Omega) = \phi_G(\Lambda_0, \Omega_0), \Lambda \in \mathbb{R}^D, \Omega \in \mathrm{PD}(B)\}.$$

A fiber contains all pairs of matrices that induce the same observed covariance matrix Σ. For $\Sigma \in \operatorname{im} \phi_G$, we also write $\mathcal{F}_G(\Sigma)$ for the fiber belonging to Σ. We can phrase identifiability in terms of fibers:

- G is globally identifiable, if $|\mathcal{F}_G(\Lambda_0, \Omega_0)| = 1$ for all $\Lambda_0 \in \mathbb{R}^D$ and $\Omega_0 \in \mathrm{PD}(B)$.
- G is generically identifiable, if $|\mathcal{F}_G(\Lambda_0, \Omega_0)| = 1$ for Zariski almost all $\Lambda_0 \in \mathbb{R}^D$ and $\Omega_0 \in \mathrm{PD}(B)$.

Generic identifiability asks whether all parameters are almost always identifiable in the Zariski sense, that is, everywhere except for a lower dimensional algebraic set. For generic identifiability, we only consider the parameters $\lambda_{i,j}$ since the parameters $\omega_{k,l}$ can be recovered from the $\lambda_{i,j}$ and $\sigma_{i,j}$ using (9.1), see also [Drt18].

Global and generic identifiability are properties of the given mixed graph. In this work, we also study identification as a property of the observed numerical data, i.e., of the observed covariance matrix Σ.

9.3 Definition (Numerical Identifiability)**.** Given an acyclic mixed graph $G = (V, D, B)$ and a feasible matrix Σ, decide whether the parameters are uniquely identifiable, i.e. if $|\mathcal{F}_G(\Sigma)| = 1$?

Note that this is a promise problem. We assume that Σ is feasible, i.e., in the image of ϕ_G. Therefore, we shall also study the feasibility problem: Given Σ, is it contained in $\operatorname{im} \phi_G$?

9.3 Finding Another Solution

Numerical identification is a promise problem, i.e., we assume that the given input is feasible. Being a promise problem means that an algorithm for numerical identification should output the correct answer whenever the input is feasible. But it can output anything when the input is not feasible. See Section 2.6 for some further information about promise problems.

For our hardness proof, we need to look at instances of ETR or QUAD that are satisfiable. Of course, deciding whether a satisfiable instance is satisfiable is a trivial task. So the task will be to decide whether the satisfiable instance has another solution. We call the corresponding promise problems ETR^{++} and QUAD^{++}.

It turns out that these promise problems are $\exists\mathbb{R}$-hard. Since QUAD^{++} is a special case of ETR^{++}, it suffices to prove this for QUAD^{++}. Let $p_1, \dots, p_m$ be a QUAD instance in the variables $x_1, \dots, x_n$ and let y be an extra variable. We will plant an extra solution into the system:

$$y(y-1) = 0 \tag{9.4}$$
$$yx_i = 0 \qquad i = 1, \dots, n \tag{9.5}$$
$$(y-1)p_j = 0 \qquad j = 1, \dots, m \tag{9.6}$$

9.7 Lemma. *The system above has the following solutions:*

1. $y = 1$, $x_1 = \cdots = x_n = 0$
2. $y = 0$, $x_1 = \xi_1, \dots, x_n = \xi_n$, *where* $\xi \in \mathbb{R}^n$ *is any solution to the original instance.*

In particular, the system always has a solution. It has more than one solution iff the original QUAD*-instance is satisfiable.*

Proof. The first equation (9.4) constrains y to be $\{0,1\}$-valued. If $y = 0$, then the equations (9.5) are trivially satisfied and (9.6) reduces to the original instance (2.5). If $y = 1$, then the equations (9.6) are trivially satisfied and (9.5) reduces to $x_1 = \cdots = x_n = 0$. Note that in both cases we always get different solutions since the y-value differs. □

While the resulting instance has polynomials of degree 3, we bring all polynomials back to degree 2 and the special form required in the definition of QUAD (see Equation (2.6)) by using the same trick due to Tseitin used in Lemma 2.7. Tseitin's trick preserves the number of solutions exactly: all added variables are uniquely determined once $x_1, \dots, x_n$ and y are fixed. Using the transformation in the lemma above, together with Tseitin's trick, we can map any QUAD-instance into a QUAD^{++}-instance and obtain

9.8 Corollary. ETR^{++} *and* QUAD^{++} *are* $\exists\mathbb{R}$*-hard.*

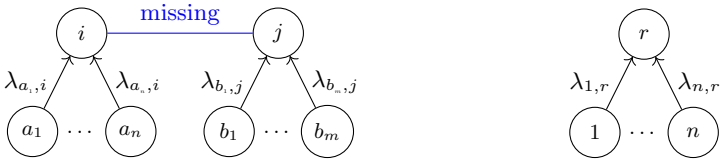


Figure 9.3: Left: A single missing edge on the top layer. Right: The gadget storing the value of each variable. $\lambda_{i,r}$ corresponds to x_i.

9.4 Hardness of Numerical Identifiability

This section is dedicated to proving:

9.9 Theorem. *Numerical identifiability is $\forall\mathbb{R}$-hard.*

The proof consists of building a polynomial-time reduction from the complement of QUAD^{++} to numerical identifiability, i.e., we construct an acyclic mixed graph G and a $\Sigma \in \operatorname{im}\phi_G$, such that the fiber $\mathcal{F}_G(\Sigma)$ has size 1 iff the given QUAD^{++}-instance has only one solution. For this, we use the following characterization of fibers due to [Drt18]:

9.10 Lemma. *Let $G = (V, D, B)$ be an acyclic mixed graph, and let $\Sigma \in \operatorname{im}\phi_G$. The fiber $\mathcal{F}_G(\Sigma)$ is isomorphic to the set of matrices $\Lambda \in \mathbb{R}^D$ that solve the equation system*

$$[(I-\Lambda)^T\Sigma(I-\Lambda)]_{i,j} = 0, \qquad i \neq j, i \leftrightarrow j \notin B \tag{9.11}$$

We construct G as follows: The directed edges form a bipartite graph with edges going from the bottom layer to the top layer. Every node at the bottom layer has outdegree one. Moreover bidirected edges exist between all pairs of nodes, except for certain pairs of nodes of the top layer. See Figure 9.3 (left-hand side) for an illustration.

This missing edge in Figure 9.3 corresponds to the equation

$$0 = \sigma_{i,j} - \sum_{\ell=1}^{n}\sigma_{a_\ell,j}\lambda_{a_\ell,i} - \sum_{k=1}^{m}\sigma_{b_k,i}\lambda_{b_k,j} + \sum_{\ell=1}^{n}\sum_{k=1}^{m}\sigma_{a_\ell,b_k}\lambda_{a_\ell,i}\lambda_{b_k,j} \tag{9.12}$$

in Lemma 9.10.

9.13 Observation. All σ values that appear in (9.12) cannot appear in any other missing edge equation of missing edges in the top layer. Parameters $\sigma_{a_\ell,j}$ can only appear in an equation of a missing edge that contains the node j and another node h such that there is a directed edge from a_ℓ to h. However, (a_ℓ, i) is the only such edge, since the nodes in the bottom layer only have outdegree one. The same is true for $\sigma_{b_k,j}$. σ_{a_ℓ,b_ℓ} can only appear in an equation of a missing edge $h \leftrightarrow h'$ if there are directed edges (a_ℓ, h) and (b_k, h'). By the same argument, $i \leftrightarrow j$ is the only such missing edge. Furthermore $\sigma_{i,j}$ can obviously only appear in this equation.

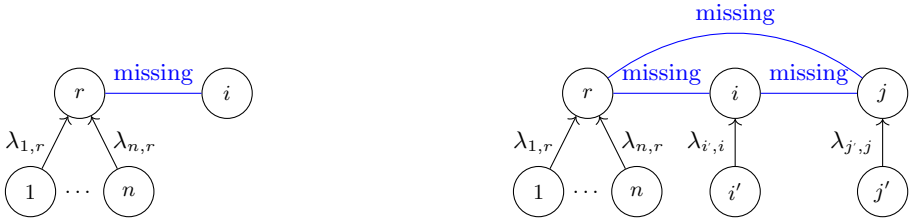


Figure 9.4: Left: Gadget for affine linear constraints. Right: Gadget for multiplicative constraints.

The above observation means that we can freely "program" the equations, that is, we can freely choose the σ-values in each missing edge equation without interfering with any other missing edge equation.

We start with a gadget with one node r in the top layer and n nodes in the bottom layer connected to it. It is used to store the value of each variable of our ETR instance, $\lambda_{1,r}$ corresponds to x_1, $\lambda_{2,r}$ corresponds to x_2, etc, see Figure 9.3 (right-hand side) for an illustration.

By assuming all polynomials in our QUAD^{++}-instance are of the forms (2.6), we need to be able to encode products and affine linear forms. We start by showing how to encode an arbitrary affine linear constraint $\sum_{\ell=1}^{n} \alpha_\ell x_\ell = \beta$ using a single additional node i in the top layer, "connected" to r via a missing edge as in Figure 9.4.

Setting $\sigma_{r,i} = \beta$ and $\sigma_{\ell,i} = \alpha_\ell$, $1 \leq \ell \leq n$, makes (9.12) together with $\lambda_{\ell,r} = x_\ell$ directly equivalent to $\sum_{\ell=1}^{n} \alpha_\ell x_\ell = \beta$.

Encoding a product $x_a = x_b \cdot x_c$ requires two additional nodes i and j in the top layer, with missing bidirectional edges between them and r. Furthermore we introduce two nodes i' and j' in the bottom layer, connected to i and j respectively, see Figure 9.4 (right-hand side). This introduces three equations. The missing edge $r \leftrightarrow i$ enforces $\lambda_{i',i} = \lambda_{c,r}$ by setting $\sigma_{r,i} = \sigma_{1,i'} = \ldots = \sigma_{n,i'} = 0$, $\sigma_{r,i'} = -1$, $\sigma_{c,i} = 1$, and $\sigma_{\ell,i} = 0$, for all $\ell \in \{1, \ldots, n\} \setminus \{c\}$ in (9.12). We use the missing edge $i \leftrightarrow j$ to further ensure $\lambda_{j',j} = \lambda_{i',i} = \lambda_{c,r}$, for which we set $\sigma_{i,j} = \sigma_{i',j'} = 0$, $\sigma_{i',j} = 1$, and $\sigma_{i,j'} = -1$. After having copied $\lambda_{c,r}$ twice, we are finally able to enforce the multiplication itself using the missing edge $r \leftrightarrow j$. We set $\sigma_{r,j} = \sigma_{r,j'} = 0$, $\sigma_{a,j} = 1$, $\sigma_{b,j'} = 1$, $\sigma_{\ell,j} = 0$, for all $\ell \in \{1, \ldots, n\} \setminus \{a\}$, and $\sigma_{\ell,j'} = 0$, for all $\ell \in \{1, \ldots, n\} \setminus \{b\}$. We need to copy the parameter $\lambda_{c,r}$ twice to be able to "program" the equation corresponding to the missing edge $r \leftrightarrow j$.

Proof of Theorem 9.9. Let polynomials $p_1, \ldots, p_m$ in variables $x_1, \ldots, x_n$ be a QUAD^{++}-instance with all polynomials being one of the forms in (2.6). Let the number of affine linear polynomials among $p_1, \ldots, p_m$ be k. Then the graph $G = (V, D, B)$ constructed above has $\ell := 1 + n + k + 4(m-k) = 1 + n + 4m - 3k$ nodes. Using Observation 9.13, we see that the construction induces a well-defined partial matrix $\Sigma \in \mathbb{R}^{\ell \times \ell}$. Every entry

of Σ not defined by the construction is set to 0 if it is off-diagonal and ℓ if it is on the diagonal. Since all $\sigma_{i,j}$ set in the construction are from $\{-1, 0, 1\}$ and off-diagonal, Σ is strictly diagonally dominant by our choice of ℓ and thus positive definite by the Gershgorin circle theorem [Ger31].

Remains to prove $\Sigma \in \operatorname{im} \phi_G$. Let $\xi \in \mathbb{R}^n$ be any solution with $p_1(\xi) = \cdots = p_m(\xi) = 0$. The existence of ξ is guaranteed by the promise of QUAD^{++}. Create $\Lambda \in \mathbb{R}^{\ell \times \ell}$ as follows: $\lambda_{i,r} = \xi_i$ for $i \in \{1, \ldots, n\}$ and $\lambda_{i',i} = \lambda_{j',j} = \xi_c$ whenever the vertices i, i', j, j' are the vertices added by the construction due to a multiplication. All other entries of Λ are 0. Then $I - \Lambda$ is invertible and we have $\Sigma = \phi_G(\Lambda, \Omega)$ for $\Omega = (I - \Lambda)^T \Sigma (I - \Lambda)$. Furthermore Ω is positive definite due to Σ being positive definite and $\Omega \in \mathrm{PD}(B)$.

By Lemma 9.10, this implies that $|\mathcal{F}_G(\Sigma)|$ is precisely the number of solutions of our QUAD^{++}-instance. So if the QUAD^{++}-instance is a yes-instance, that is, has more than one solution, then our constructed instance is not numerically identifiable. If the QUAD^{++}-instance is a no-instance, that is, has only one solution, then our constructed instance is numerically identifiable. So we have a reduction from the complement of QUAD^{++}. The theorem now follows, since by Corollary 9.8, QUAD^{++} is $\exists\mathbb{R}$-hard and the complement of an $\exists\mathbb{R}$-hard problem is $\forall\mathbb{R}$-hard. □

9.5 Upper Bound for Numerical Identifiability

In this section, we show a $\forall\mathbb{R}$ upper bound for numerical identifiability and thus, combined with Theorem 9.9, prove an almost[1] matching lower and upper bound. We start with the following lemma. The $\exists\mathbb{R}$ part will be needed in the next section.

[1] see Remark 9.20 for the details.

9.14 Lemma. *Membership in* $\mathrm{PD}(n)$ *and* $\mathrm{PD}(B)$ *can be expressed in* $\exists\mathbb{R}$ *and* $\forall\mathbb{R}$. [2]

[2] Note that membership in $\mathrm{PD}(n)$ can be decided even faster. However, this will not change the complexity of our overall algorithm.

Proof. For the $\exists\mathbb{R}$ expression, we use the fact that every real positive definite matrix $A \in \mathbb{R}^{n \times n}$ has a Cholesky decomposition $A = LL^T$ where L is a real lower triangular matrix with positive diagonal entries. We can thus express $A \in \mathrm{PD}(n)$ as

$$\exists L \in \mathbb{R}^{n \times n} : A = LL^T \wedge \bigwedge_{i \in \{1,\ldots,n\}} (L_{i,i} > 0 \quad \wedge \bigwedge_{j \in \{i+1,\ldots,n\}} L_{i,j} = 0). \tag{9.15}$$

We quantify over matrices and consider matrix equations in (9.15). But this can be easily rewritten as an ETR-instance by quantifying over all entries of the matrix and having one individual equation for each entry of the matrix equation.

For the $\forall\mathbb{R}$ expression, we directly use the definition of positive definite matrices to express $A \in \mathrm{PD}(n)$ as

$$\forall x \in \mathbb{R}^n : x \neq 0 \implies x^T A x > 0. \tag{9.16}$$

For membership $A \in \mathrm{PD}(B)$, in both $\exists\mathbb{R}$ and $\forall\mathbb{R}$, we add the constraint

$$\bigwedge_{(i,j)\notin B \wedge i \neq j} A_{i,j} = 0 \tag{9.17}$$

to (9.15) and (9.16), respectively. □

We remind the reader that numerical identifiability is a promise problem with the promise that the input $\Sigma \in \operatorname{im}\phi_G$, so it suffices to check whether all elements in the fiber $\mathcal{F}_G(\Sigma)$ are identical.

$$\begin{aligned}\forall \Lambda_1, \Lambda_2 \in \mathbb{R}^D, &\Omega_1, \Omega_2 \in \mathrm{PD}(B) : \\ &\phi_G(\Lambda_1, \Omega_1) = \phi_G(\Lambda_2, \Omega_2) = \Sigma \Rightarrow (\Lambda_1 = \Lambda_2 \wedge \Omega_1 = \Omega_2).\end{aligned} \tag{9.18}$$

The checks $\phi_G(\Lambda_i, \Omega_i) = \Sigma$ are implemented using $\Omega_i = (I - \Lambda_i)^T \Sigma (I - \Lambda_i)$, which is equivalent due to $I - \Lambda_i$ being invertible for any $\Lambda_i \in \mathbb{R}^D$. This proves the following:

9.19 Theorem. *Numerical identifiability is in (the promise version of)* $\forall\mathbb{R}$.

9.20 Remark. Strictly speaking, numerical identifiability is not contained in $\forall\mathbb{R}$, since it is a promise problem, that is, the outcome is not specified for Σ that are not feasible. $\forall\mathbb{R}$ consists by definition only of classical decision problems, where the outcome is specified for *all* inputs. So the corresponding complexity class is Pr-$\forall\mathbb{R}$.

However, we can express feasibility in $\exists\mathbb{R}$:

9.21 Lemma. *Membership in* $\operatorname{im}\phi_G$ *can be expressed in* $\exists\mathbb{R}$.

Proof. We use the expression $\exists \Lambda \in \mathbb{R}^D, \Omega \in \mathrm{PD}(B) : (I - \Lambda)^T \Sigma (I - \Lambda) = \Omega$, where we use Lemma 9.14 to express $\Omega \in \mathrm{PD}(B)$ in $\exists\mathbb{R}$, that is, we quantify over an arbitrary matrix Ω first and add the ETR expression from Lemma 9.14 to ensure that Ω is in $\mathrm{PD}(B)$. □

Hence, we can check in $\exists\mathbb{R}$ whether the input Σ is feasible and then in $\forall\mathbb{R}$ whether the fiber has only one element. Using Renegar's algorithm, we get:

9.22 Corollary. *Numerical identifiability can be decided in polynomial space.*

9.6 Generic Identifiability is in PSPACE

Let DIM denote the following problem: Given an encoding of a semi-algebraic set S and a number d, decide whether $\dim S \geq d$.

9.23 Lemma (Koiran [Koi99][3])**.** *The problem* DIM *is $\exists\mathbb{R}$-complete. Moreover, this is even true when the set is given by an existentially quantified formula as in (2.4).*

We use the same notation as in Section 9.2.1. Let $G = (V, D, B)$ be a mixed graph. Let $S_G = \{(\Lambda, \Omega) \mid |\mathcal{F}_G(\Lambda, \Omega)| > 1, \Lambda \in \mathbb{R}^D, \Omega \in \mathrm{PD}(B)\}$.

[3]Koiran mainly works in the so-called BSS-model of real computation. In this model, one is also allowed to use arbitrary real constants in the algorithm as well as real inputs. In $\exists\mathbb{R}$, we only allow the constants 0 and 1 and the inputs are given by some binary encoding. However, Koiran also considers the bit model. [Koi99, Theorem 6] proves the computational equivalence of DIM and 4FEAS in the bit model. Since 4FEAS is $\exists\mathbb{R}$-complete [SŠ17], this implies that DIM is $\exists\mathbb{R}$-complete. Note that at the time Koiran wrote his paper, the class $\exists\mathbb{R}$ was not formally defined and therefore, Koiran does not mention it explicitly. For the moreover part, note that [Koi99, Section 1.1] discusses the representations of semi-algebraic sets that are supported by his proof. There he mentions existentially quantified formulas explicitly.

9.24 Observation. G is generically identifiable iff $\dim \mathbb{R}^D + \dim \mathrm{PD}(B) > \dim S_G$.

Proof. As $S_G \subseteq \mathbb{R}^D \times \mathrm{PD}(B)$ and $\dim(\mathbb{R}^D \times \mathrm{PD}(B)) = \dim \mathbb{R}^D + \dim \mathrm{PD}(B)$, the right-hand side is just the definition of being generically identifiable. □

The dimensions of $\mathbb{R}^D$ and $\mathrm{PD}(B)$ are easy to calculate:

9.25 Lemma. $\dim \mathbb{R}^D = |D|$ *and* $\dim \mathrm{PD}(B) = n + |B|$.

Proof. By definition, $\mathbb{R}^D$ are just matrices with $|D|$ degrees of freedom, directly giving $\dim \mathbb{R}^D = |D|$. For the dimension of $\mathrm{PD}(B)$, remember that real positive definite matrices are always symmetric. The bidirected edges in B thus allow for $|B|$ degrees of freedom. Furthermore, the Gershgorin circle theorem [Ger31] implies that, no matter the values off the diagonal, choosing the diagonal values large enough makes a matrix positive definite, allowing for an additional n degrees of freedom. □

We postpone the proof that membership in S_G (and thus its dimension) can be expressed in $\exists\mathbb{R}$, in favor of first giving our algorithm to decide generic identifiability, using Observation 9.24:

9.26 Theorem. *Generic identifiability is in $\forall\mathbb{R}$.*

Proof. Let G be the given mixed graph. Formulate membership in S_G as an instance of ETR using Lemma 9.28. Note that the number of variables and the size of this instance is polynomial in the size of G. Now we can check whether G is generically identifiable by checking the condition in Observation 9.24 by using Koiran's algorithm (see Lemma 9.23) to check if $\dim \mathbb{R}^D + \dim \mathrm{PD}(B) = |B| + |D| + n > \dim S_G$ and accepting if this is the case and rejecting otherwise.

Note that the check $\dim S_G < |B| + |D| + n$ needs to be implemented as $\neg(\dim S_G \geq |B| + |D| + n)$, thus being in $\forall\mathbb{R}$ by De Morgan's laws. □

Using Renegar's algorithm this implies:

9.27 Corollary. *Generic identifiability can be decided in* PSPACE.

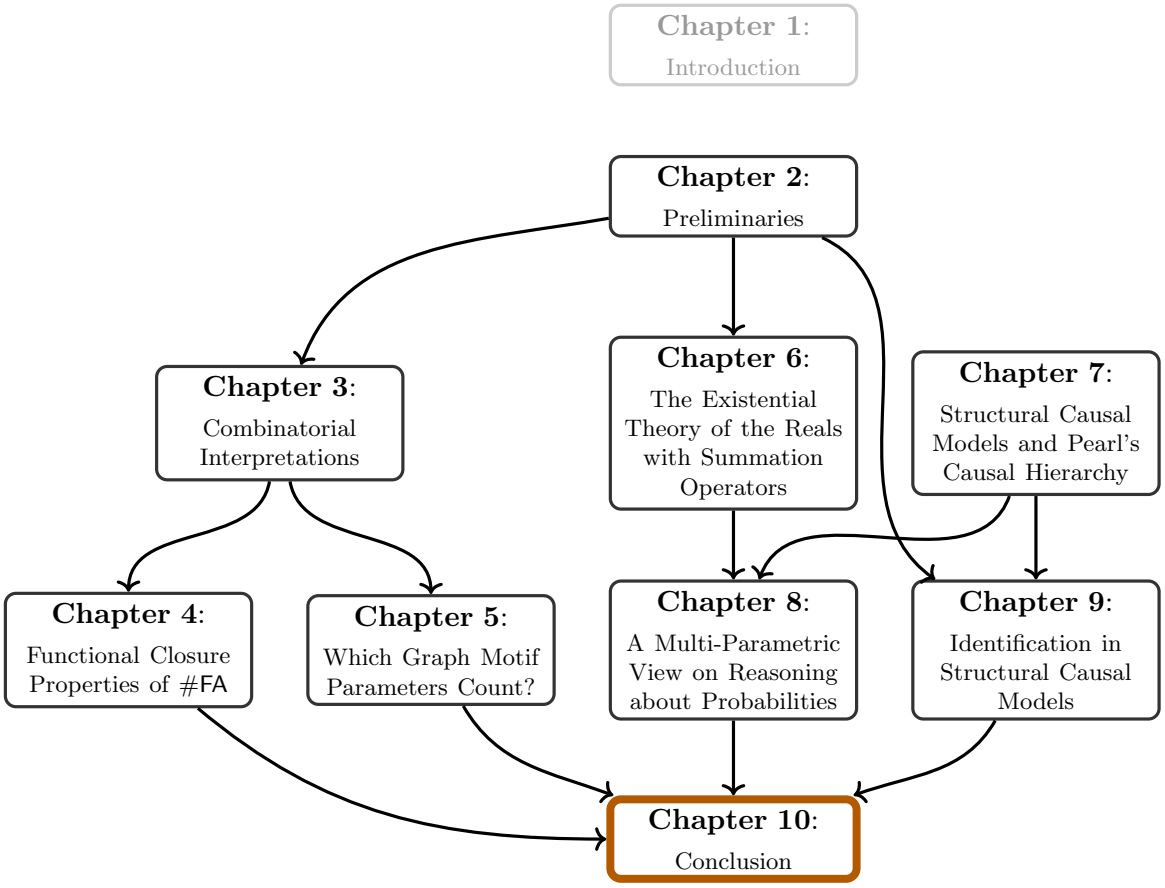
It only remains to show how to express membership in S_G as an ETR-formula.

9.28 Lemma. *Membership in S_G can be expressed in $\exists\mathbb{R}$.*

Proof. The membership of some (Λ, Ω) in S_G can be expressed as

$$\begin{aligned}\Lambda \in \mathbb{R}^D \wedge \Omega \in \mathrm{PD}(B) \wedge \exists \Sigma \in \mathbb{R}^{n\times n}, \Lambda' \in \mathbb{R}^D, \Omega' \in \mathrm{PD}(B) : (I-\Lambda)^T\Sigma(I-\Lambda) = \Omega \\ \wedge (I-\Lambda')^T\Sigma(I-\Lambda') = \Omega' \\ \wedge (\Lambda \neq \Lambda' \vee \Omega \neq \Omega')\,.\end{aligned}$$

□


Chapter 1:
Introduction
Chapter 2:
Preliminaries
Chapter 3:
Combinatorial Interpretations
Chapter 6:
The Existential Theory of the Reals with Summation Operators
Chapter 7:
Structural Causal Models and Pearl's Causal Hierarchy
Chapter 4:
Functional Closure Properties of #FA
Chapter 5:
Which Graph Motif Parameters Count?
Chapter 8:
A Multi-Parametric View on Reasoning about Probabilities
Chapter 9:
Identification in Structural Causal Models
Chapter 10:
Conclusion

Conclusion

On our quest to find the correct machine model for combinatorial interpretability, we have seen in Chapter 4 that the univariate functional closure properties of #FA are precisely the ultimately PORC functions and the multivariate functional closures are just sums and products thereof. Unfortunately, since this includes some modular arithmetic, divisions and subtractions, it allows for operations generally considered to not be combinatorial. The likely next models to investigate would be pushdown automata or linear bounded automata. On the other hand, we've seen in Chapter 5 that local combinatorial interpretations play nicely with (the promise version of) $\#\mathsf{P}^{\vdots}$, opening up future research to treat oracle inputs as first class citizens in the pursuit of combinatorial interpretability instead of just an unwanted artifact.

Then, in Chapter 6 we saw that augmenting the existential theory of the reals with unary summations increases the complexity of deciding whether a given sentence is true from $\mathsf{NP}_{\text{real}}$ without unary summations to $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ for unary summations without variable indexing and all the way to $\mathsf{NEXP}_{\text{real}}$ when the summation variables can be used to index variables. Natural complete problems for $\mathsf{NP}_{\text{real}}$ are ubiquitous (see [SCM24]) and we have seen some complete problems for $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ and $\mathsf{NEXP}_{\text{real}}$ in Chapter 8. A natural question for further research is now, how common are these complexity classes? Candidates for $\mathsf{NEXP}_{\text{real}}$-completeness are problems where there is a known gap in knowledge between NEXP-hardness and containment in EXPSPACE. On the other hand, candidates for $\mathsf{NP}_{\text{real}}^{\mathsf{VNP}_{\mathbb{R}}}$ are problems that are between the polynomial time hierarchy PH and PSPACE.

Moreover, the precise relationships of these classes compared to more "classical" complexity classes, defined on Turing machines, remains wide open. While we expect all inclusions $\mathsf{NP} \subseteq \mathsf{NP}_{\text{real}} \subseteq \mathsf{PSPACE}$ and $\mathsf{NEXP} \subseteq \mathsf{NEXP}_{\text{real}} \subseteq \mathsf{EXPSPACE}$ to be strict, it is unclear when exactly real computations can be replaced by Boolean ones. For example we observed that Π-ETR (the existential theory of the reals with an added unary product, but without variable indexing) is PSPACE-complete in Chapter 6 and the complex landscape in Chapter 8 regularly switches between Boolean and real classes.

In Chapter 9 we learned that numerical identification is $\forall\mathbb{R}$-hard and can be solved in the promise version of $\forall\mathbb{R}$. While this rules out efficient general complete and sound algorithms, the reduction only shows hardness for mixed graphs with two layers (relative to the directed edges) and very dense ($|V|^2 - O(|V|)$ many) bidirectional edges. In particular, mixed graphs of this form don't really occur in real data, leaving open the question of which graph classes still allow for efficient identification algorithms. There has been some progress on this, with [ZWBL22; GB24] showing identification for mixed graphs where the directed edges form a rooted tree can be solved in randomized polynomial time, although only generically. Even though we have only shown $\forall\mathbb{R}$-hardness for numerical identification, we also conjecture the same hardness for generic identification, matching the $\forall\mathbb{R}$ upper bound we've shown.

Bibliography

[AKH] Scott Aaronson, Greg Kuperberg, and Oliver Habryka. *The Complexity Zoo*. `https://complexityzoo.net/`. Accessed: 2025-12-03.

[AAM18] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. "The art gallery problem is $\exists\mathbb{R}$-complete". In: *Proc. ACM SIGACT Symposium on Theory of Computing (STOC)*. 2018, pp. 65–73.

[Ach15] Antonis Achilleos. "NEXP-completeness and universal hardness results for justification logic". In: *Proc. International Computer Science Symposium in Russia*. Springer. 2015, pp. 27–52.

[AR56] T.W. Anderson and Herman Rubin. "Statistical inference in factor analysis". In: *Proc. of the Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press. 1956, p. 111.

[BCII22] Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. "On Pearl's Hierarchy and the Foundations of Causal Inference". In: *Probabilistic and Causal Inference: The Works of Judea Pearl*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 507–556.

[BPR03] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2003.

[Bea+95] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. "The relative complexity of NP search problems". In: *Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing*. 1995, pp. 303–314.

[BIP95] Paul Beame, Russell Impagliazzo, and Toniann Pitassi. "Improved Depth Lower Bounds for Small Distance Connectivity". In: *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*. 1995, pp. 692–701. DOI: `10.1109/SFCS.1995.492671`.

[BR07] Matthias Beck and Sinai Robins. *Computing the continuous discretely*. Vol. 61. Springer, 2007.

[BB61] Edwin F Beckenbach and Richard Bellman. *Inequalities*. Springer, Berlin, 1961.

[Bei97] Richard Beigel. "Closure properties of GapP and #P". In: *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*. IEEE. 1997, pp. 144–146. DOI: `10.1109/ISTCS.1997.595166`.

[Ber+23] Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, and Simon Weber. "Training Fully Connected Neural Networks is $\exists\mathbb{R}$-Complete". In: *Proc. Conference on Neural Information Processing Systems (NeurIPS)*. 2023.

[BCDI25] Markus Bläser, Radu Curticapean, Julian Dörfler, and Christian Ikenmeyer. "Which Graph Motif Parameters Count?" In: *50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, Warsaw, Poland, August 25-29, 2025*. Vol. 345. LIPIcs. 2025, 23:1–23:18. DOI: 10.4230/LIPICS.MFCS.2025.23.

[BDLZ24] Markus Bläser, Julian Dörfler, Maciej Liśkiewicz, and Benito van der Zander. "The Existential Theory of the Reals with Summation Operators". In: *Proc. 35th International Symposium on Algorithms and Computation (ISAAC)*. 2024.

[BDLZ25] Markus Bläser, Julian Dörfler, Maciej Liśkiewicz, and Benito van der Zander. "Probabilistic and Causal Satisfiability: Constraining the Model". In: *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*. Vol. 334. LIPIcs. 2025, 144:1–144:20. DOI: 10.4230/LIPICS.ICALP.2025.144.

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. "On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines". In: *Bulletin of the American Mathematical Society* 21.1 (1989), pp. 1–46.

[BCR13] J. Bochnak, M. Coste, and M.F. Roy. *Real Algebraic Geometry*. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics. Springer Berlin Heidelberg, 2013. ISBN: 9783662037188.

[Bol89] Kenneth A. Bollen. *Structural equations with latent variables*. John Wiley & Sons, 1989.

[BT84] R.J. Bowden and D.A. Turkington. *Instrumental variables*. Cambridge University Press, 1984.

[Bri10] Carlos Brito. "Instrumental sets". In: *Heuristics, Probability and Causality. A Tribute to Judea Pearl*. Ed. by Rina Dechter, Hector Geffner, and Joseph Y Halpern. College Publications, 2010. Chap. 17, pp. 295–308.

[BP02a] Carlos Brito and Judea Pearl. "A graphical criterion for the identification of causal effects in linear models". In: *Proc. AAAI Conference on Artificial Intelligence (AAAI)*. 2002, pp. 533–538.

[BP02b] Carlos Brito and Judea Pearl. "Generalized Instrumental Variables". In: *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*. 2002, pp. 85–93.

[Bro84] Andrei Z Broder. "The r-Stirling numbers". In: *Discrete Mathematics* 49.3 (1984), pp. 241–259.

[Bro74] Michael W. Browne. "Generalized least squares estimators in the analysis of covariance structures". In: *South African Statistical Journal* 8.1 (1974), pp. 1–24.

[Bür00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory.* Vol. 7. Springer Science & Business Media, 2000.

[Cai+89] Jin-Yi Cai, Thomas Gundermann, Juris Hartmanis, Lane A Hemachandra, Vivian Sewelson, Klaus Wagner, and Gerd Wechsung. "The boolean hierarchy II: Applications". In: *SIAM Journal on Computing* 18.1 (1989), pp. 95–111.

[CP24] Swee Hong Chan and Igor Pak. "Computational complexity of counting coincidences". In: *Theoretical Computer Science* 1015 (2024), p. 114776.

[Che16] Bryant Chen. "Identification and overidentification of linear structural equation models". In: *Proc. International Conference on Neural Information Processing Systems (NeurIPS).* 2016, pp. 1587–1595.

[CKB17] Bryant Chen, Daniel Kumor, and Elias Bareinboim. "Identification and model testing in linear structural equation models using auxiliary variables". In: *Proc. International Conference on Machine Learning (ICML).* PMLR. 2017, pp. 757–766.

[CPB15] Bryant Chen, Judea Pearl, and Elias Bareinboim. "Incorporating knowledge into structural equation models using auxiliary variables". In: *Proc. International Joint Conference on Artificial Intelligence (IJCAI).* 2015, pp. 3577–3583.

[Coo72] Stephen A Cook. "A hierarchy for nondeterministic time complexity". In: *Proceedings of the fourth annual ACM symposium on Theory of computing.* 1972, pp. 187–192.

[Coo90] Gregory F Cooper. "The computational complexity of probabilistic inference using Bayesian belief networks". In: *Artificial Intelligence* 42.2-3 (1990), pp. 393–405.

[Cur24] Radu Curticapean. "Count on CFI graphs for #P-hardness". In: *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024.* 2024, pp. 1854–1871. DOI: 10.1137/1.9781611977912.74.

[CDM17] Radu Curticapean, Holger Dell, and Dániel Marx. "Homomorphisms are a good basis for counting small subgraphs". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada.* 2017, pp. 210–223. DOI: 10.1145/3055399.3055502.

[DL93] Paul Dagum and Michael Luby. "Approximating probabilistic inference in Bayesian belief networks is NP-hard". In: *Artificial Intelligence* 60.1 (1993), pp. 141–153.

[DI24] Julian Dörfler and Christian Ikenmeyer. "Functional Closure Properties of Finite ℕ-Weighted Automata". In: *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*. Vol. 297. LIPIcs. 2024, 134:1–134:18. DOI: 10.4230/LIPICS.ICALP.2024.134.

[DRSW22] Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. "Counting Induced Subgraphs: An Algebraic Approach to #W[1]-Hardness". In: *Algorithmica* 84.2 (2022), pp. 379–404. DOI: 10.1007/S00453-021-00894-9.

[DZBL24] Julian Dörfler, Benito van der Zander, Markus Bläser, and Maciej Liśkiewicz. "On the Complexity of Identification in Linear Structural Causal Models". In: *Proc. 38th Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2024.

[DZBL25] Julian Dörfler, Benito van der Zander, Markus Bläser, and Maciej Liśkiewicz. "From Probability to Counterfactuals: the Increasing Complexity of Satisfiability in Pearl's Causal Hierarchy". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. 2025.

[DMW24] Simon Döring, Dániel Marx, and Philip Wellnitz. "Counting Small Induced Subgraphs with Edge-Monotone Properties". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. 2024, pp. 1517–1525. DOI: 10.1145/3618260.3649644.

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

[DK21] Manfred Droste and Dietrich Kuske. "Weighted automata". In: *Handbook of Automata Theory*. Ed. by Jean-Éric Pin. European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 113–150. DOI: 10.4171/Automata-1/4.

[Drt18] Mathias Drton. "Algebraic problems in structural equation modeling". In: *The 50th anniversary of Gröbner bases*. Mathematical Society of Japan, 2018, pp. 35–86.

[DFS11] Mathias Drton, Rina Foygel, and Seth Sullivant. "Global identifiability of linear structural equation models". In: *The Annals of Statistics* 39.2 (2011), pp. 865–886.

[DM17] Mathias Drton and Marloes H Maathuis. "Structure learning in graphical modeling". In: *Annual Review of Statistics and Its Application* 4 (2017), pp. 365–393.

[Dun75] Otis Dudley Duncan. *Introduction to structural equation models*. Academic Press, 1975.

[EHM24] Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. "Smoothing the Gap Between NP and ER". In: *SIAM J. Comput.* 53.6 (2024), S20–102. DOI: 10.1137/20M1385287.

[FHM90] Ronald Fagin, Joseph Y Halpern, and Nimrod Megiddo. "A logic for reasoning about probabilities". In: *Information and Computation* 87.1-2 (1990), pp. 78–128.

[Fis66] Franklin M. Fisher. *The identification problem in econometrics*. McGraw-Hill, 1966.

[Fis36] Ronald Aylmer Fisher. "Design of experiments". In: *British Medical Journal* 1.3923 (1936), p. 554.

[FW78] Steven Fortune and James Wyllie. "Parallelism in random access machines". In: *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*. STOC '78. San Diego, California, USA, 1978, pp. 114–118. ISBN: 9781450374378. DOI: 10.1145/800133.804339.

[FDD12] Rina Foygel, Jan Draisma, and Mathias Drton. "Half-trek criterion for generic identifiability of linear structural equation models". In: *The Annals of Statistics* 40.3 (2012), pp. 1682–1713.

[GSS10] Luis D. García-Puente, Sarah Spielvogel, and Seth Sullivant. "Identifying Causal Effects with Computer Algebra". In: *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press. 2010, pp. 193–200.

[GS00] Mariano Gasca and Thomas Sauer. "Polynomial interpolation in several variables". In: *Advances in Computational Mathematics* 12 (2000), pp. 377–410.

[Ger31] S Geršgorin. "Über die Abgrenzung der Eigenwerte einer Matrix". In: *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* 6 (1931), pp. 749–754.

[GZS19] Clark Glymour, Kun Zhang, and Peter Spirtes. "Review of causal discovery methods based on graphical models". In: *Frontiers in genetics* 10 (2019), p. 524.

[GLR72] RL Graham, K Leeb, and BL Rothschild. "Ramsey's theorem for a class of categories". In: *Advances in Mathematics* 8.3 (1972), pp. 417–433.

[GV88] Dima Grigoriev and Nicolai Vorobjov. "Solving systems of polynomial inequalities in subexponential time". In: *J. Symb. Comput.* 5.1/2 (1988), pp. 37–64.

[GB24] Aaryan Gupta and Markus Bläser. "Identification for Tree-Shaped Structural Causal Models in Polynomial Time". In: *Proc. AAAI Conference on Artificial Intelligence (AAAI)*. 2024, pp. 20404–20411.

[HLP52] HG Hardy, JE Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1952.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2002. ISBN: 978-3-642-08684-7. DOI: 10.1007/978-3-662-04880-1.

[Her95] Ulrich Hertrampf. "Classes of bounded counting type and their inclusion relations". In: *STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science Munich, Germany, March 2–4, 1995 Proceedings 12*. Springer. 1995, pp. 60–70.

[HVW95] Ulrich Hertrampf, Heribert Vollmer, and Klaus W Wagner. "On the power of number-theoretic operations with respect to counting". In: *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*. IEEE. 1995, pp. 299–314. DOI: 10.1109/SCT.1995.514868.

[HBK92] Li-tze Hu, Peter M Bentler, and Yutaka Kano. "Can test statistics in covariance structure analysis be trusted?" In: *Psychological bulletin* 112.2 (1992), p. 351.

[II20] Duligur Ibeling and Thomas Icard. "Probabilistic Reasoning Across the Causal Hierarchy". In: *Proc 34th AAAI Conference on Artificial Intelligence, AAAI*. 2020, pp. 10170–10177.

[IIMM24] Duligur Ibeling, Thomas Icard, Krzysztof Mierzewski, and Milan Mossé. "Probing the Quantitative–Qualitative Divide in Probabilistic Reasoning". In: *Annals of Pure and Applied Logic* 175.9 (2024), p. 103339.

[IIM24] Duligur Ibeling, Thomas Icard, and Milan Mossé. "On probabilistic and causal reasoning with summation operators". In: *Journal of Logic and Computation* (2024), exae068.

[IP22] Christian Ikenmeyer and Igor Pak. "What is in #P and what is not?" In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 860–871. DOI: 10.1109/FOCS54457.2022.00087.

[IPP23] Christian Ikenmeyer, Igor Pak, and Greta Panova. "Positivity of the symmetric group characters is as hard as the polynomial time hierarchy". In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2023, pp. 3573–3586.

[Isb91] John Isbell. "Some inequalities in hom sets". In: *Journal of Pure and Applied Algebra* 76.1 (1991), pp. 87–110. ISSN: 0022-4049. DOI: https://doi.org/10.1016/0022-4049(91)90099-N.

[JM15a] Mark Jerrum and Kitty Meeks. "Some Hard Families of Parameterized Counting Problems". In: *ACM Trans. Comput. Theory* 7.3 (2015), 11:1–11:18. DOI: 10.1145/2786017.

[JM15b] Mark Jerrum and Kitty Meeks. "The parameterised complexity of counting connected subgraphs and graph motifs". In: *J. Comput. Syst. Sci.* 81.4 (2015), pp. 702–716. DOI: 10.1016/j.jcss.2014.11.015.

[JM17] Mark Jerrum and Kitty Meeks. "The parameterised complexity of counting even and odd induced subgraphs". In: *Comb.* 37.5 (2017), pp. 965–990. DOI: 10.1007/s00493-016-3338-5.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. "How Easy is Local Search?" In: *J. Comput. Syst. Sci.* 37.1 (1988), pp. 79–100. DOI: 10.1016/0022-0000(88)90046-3.

[Jör69] Karl G. Jöreskog. "A general approach to confirmatory maximum likelihood factor analysis". In: *Psychometrika* 34.2 (1969), pp. 183–202.

[KC02] Victor Kac and Pokman Cheung. "q-Binomial Coefficients and Linear Algebra over Finite Fields". In: *Quantum Calculus*. New York, NY: Springer New York, 2002, pp. 21–26. ISBN: 978-1-4613-0071-7. DOI: 10.1007/978-1-4613-0071-7_7.

[KR90] Richard M. Karp and Vijaya Ramachandran. "Parallel Algorithms for Shared-Memory Machines". In: *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*. Ed. by Jan van Leeuwen. Elsevier and MIT Press, 1990, pp. 869–942.

[KLMP04] Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. "Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton". In: *Theoretical Computer Science* 327.3 (2004), pp. 349–373. DOI: 10.1016/j.tcs.2004.02.049.

[Ko91] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser / Springer, 1991. ISBN: 978-1-4684-6802-1. DOI: 10.1007/978-1-4684-6802-1.

[Koi99] Pascal Koiran. "The Real Dimension Problem Is $\mathrm{NP}_{\mathbb{R}}$-Complete". In: *J. Complex.* 15.2 (1999), pp. 227–238. DOI: 10.1006/JCOM.1999.0502.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[Kos82] Carl Kostka. "Über den Zusammenhang zwischen einigen Formen von symmetrischen Functionen." Walter de Gruyter, Berlin/New York Berlin, New York. 1882.

[KCB19] Daniel Kumor, Bryant Chen, and Elias Bareinboim. "Efficient Identification in Linear Structural Causal Models with Instrumental Cutsets". In: *Proc. Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 12477–12486.

[KCB20] Daniel Kumor, Carlos Cinelli, and Elias Bareinboim. "Efficient identification in linear structural causal models with auxiliary cutsets". In: *Proc. International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 5501–5510.

[Lag24] Gregor Lagodzinski. "Counting homomorphisms over fields of prime order (Zählen von Homomorphismen über Körper mit Primzahlordnung)". PhD thesis. University of Potsdam, Germany, 2024. DOI: 10.25932/PUBLISHUP-64603.

[Lew80] Harry R. Lewis. "Complexity results for classes of quantificational formulas". In: *Journal of Computer and System Sciences* 21.3 (1980), pp. 317–353.

[LGM98] Michael L Littman, Judy Goldsmith, and Martin Mundhenk. "The computational complexity of probabilistic planning". In: *Journal of Artificial Intelligence Research* 9 (1998), pp. 1–36.

[LZY21] Ni Y Lu, Kun Zhang, and Changhe Yuan. "Improving causal discovery by optimal Bayesian network learning". In: *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 35. 10. 2021, pp. 8741–8748.

[Mac98] Saunders Mac Lane. *Categories for the Working Mathematician*. Vol. 5. Springer Science & Business Media, 1998.

[Mah14] Meena Mahajan. "Algebraic complexity classes". In: *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume* (2014), pp. 51–75.

[Mei+16] Nicolai Meinshausen, Alain Hauser, Joris M Mooij, Jonas Peters, Philip Versteeg, and Peter Bühlmann. "Methods for causal inference from gene perturbation experiments and validation". In: *Proceedings of the National Academy of Sciences* 113.27 (2016), pp. 7361–7368.

[MII22] Milan Mossé, Duligur Ibeling, and Thomas Icard. "Is Causal Reasoning Harder than Probabilistic Reasoning?" In: *The Review of Symbolic Logic* (2022), pp. 1–26.

[Mut83] Bengt Muthen. "Latent variable structural equation modeling with categorical data". In: *Journal of Econometrics* 22.1-2 (1983), pp. 43–65.

[Neš95] Jaroslav Nešetřil. "Ramsey Theory". In: *Handbook of Combinatorics*. Vol. 2. Amsterdam: Elsevier Science B.V., 1995, pp. 1331–1403. ISBN: 978-0444823465.

[NR77] Jaroslav Nešetřil and Vojtech Rödl. "Partitions of Finite Relational and Set Systems". In: *J. Comb. Theory A* 22.3 (1977), pp. 289–312. DOI: 10.1016/0097-3165(77)90004-8.

[NZZZ21] Ignavier Ng, Yujia Zheng, Jiji Zhang, and Kun Zhang. "Reliable causal discovery with improved exact search and weaker assumptions". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20308–20320.

[Nil86] Nils J Nilsson. "Probabilistic logic". In: *Artificial Intelligence* 28.1 (1986), pp. 71–87.

[OH93] Mitsunori Ogiwara and Lane A Hemachandra. "A complexity theory for feasible closure properties". In: *Journal of Computer and System Sciences* 46.3 (1993), pp. 295–325.

[Pak19] Igor Pak. "Short Stories: Combinatorial Inequalities". In: *Notices of the American Mathematical Society* 66.7 (2019), pp. 1109–1112.

[Pak23] Igor Pak. "What is a combinatorial interpretation?" In: *Open Problem in Algebraic Combinatorics* (2023).

[Pan24] Greta Panova. "Computational complexity in algebraic combinatorics". In: *Current Developments in Mathematics* 2024.1 (2024).

[Pap94] Christos H. Papadimitriou. "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence". In: *J. Comput. Syst. Sci.* 48.3 (1994), pp. 498–532. DOI: 10.1016/S0022-0000(05)80063-7.

[PD04] James D Park and Adnan Darwiche. "Complexity results and approximation strategies for MAP explanations". In: *Journal of Artificial Intelligence Research* 21 (2004), pp. 101–133.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[Pea01] Judea Pearl. *Parameter identification: A new perspective*. Tech. rep. R-276. UCLA, 2001.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, 2009, p. 464. ISBN: 0-521-77362-8.

[PM18] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[Pet72] J Peterson. "Beviser for wilsons og fermats theoremer". In: *Tidsskrift for mathematik* 2 (1872), pp. 64–65.

[RSW21a] Alexander Reisach, Christof Seiler, and Sebastian Weichwald. "Beware of the simulated dag! causal discovery benchmarks may be easy to game". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27772–27784.

[Rei+23] Alexander Reisach, Myriam Tami, Christof Seiler, Antoine Chambaz, and Sebastian Weichwald. "A Scale-Invariant Sorting Criterion to Find a Causal Order in Additive Noise Models". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 785–807.

[Ren92] James Renegar. "On the computational complexity and geometry of the first-order theory of the reals. Parts I-III". In: *Journal of symbolic computation* 13.3 (1992), pp. 255–352.

[Rie08] Emily Riehl. *Factorization Systems*. Lecture notes. 2008. URL: http://www.math.jhu.edu/~eriehl/factorization.pdf.

[Rie17] Emily Riehl. *Category Theory in Context*. Courier Dover Publications, 2017.

[Rol+22] Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. "Score matching enables causal discovery of nonlinear additive noise models". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18741–18753.

[Rot96] Dan Roth. "On the hardness of approximate reasoning". In: *Artificial Intelligence* 82.1-2 (1996), pp. 273–302.

[Rot19] Marc Roth. "Counting problems on quantum graphs". PhD thesis. Saarland University, Germany, 2019. DOI: 10.22028/D291-28348.

[RSW20] Marc Roth, Johannes Schmitt, and Philip Wellnitz. "Counting Small Induced Subgraphs Satisfying Monotone Properties". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. 2020, pp. 1356–1367. DOI: 10.1109/FOCS46700.2020.00128.

[RSW21b] Marc Roth, Johannes Schmitt, and Philip Wellnitz. "Detecting and Counting Small Subgraphs, and Evaluating a Parameterized Tutte Polynomial: Lower Bounds via Toroidal Grids and Cayley Graph Expanders". In: *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*. Vol. 198. LIPIcs. 2021, 108:1–108:16. DOI: 10.4230/LIPICS.ICALP.2021.108.

[SCM24] Marcus Schaefer, Jean Cardinal, and Tillmann Miltzow. "The Existential Theory of the Reals as a Complexity Class: A Compendium". In: *CoRR* abs/2407.18006 (2024). DOI: 10.48550/ARXIV.2407.18006. arXiv: 2407.18006.

[SS24] Marcus Schaefer and Daniel Stefankovic. "Beyond the Existential Theory of the Reals". In: *Theory Comput. Syst.* 68.2 (2024), pp. 195–226. DOI: 10.1007/S00224-023-10151-X.

[SŠ17] Marcus Schaefer and Daniel Štefankovič. "Fixed Points, Nash Equilibria, and the Existential Theory of the Reals". In: *Theory Comput. Syst.* 60.2 (2017), pp. 172–193. DOI: 10.1007/S00224-015-9662-0.

[Sch77] M.-P. Schützenberger. "La correspondance de Robinson". In: *Combinatoire et Représentation du Groupe Symétrique*. Berlin, Heidelberg, 1977, pp. 59–113. ISBN: 978-3-540-37385-8.

[Sch61] Marcel Paul Schützenberger. "On the definition of a family of automata". In: *Inf. Control.* 4.2-3 (1961), pp. 245–270. DOI: 10.1016/S0019-9958(61)80020-X.

[SFM78] Joel I Seiferas, Michael J Fischer, and Albert R Meyer. "Separating nondeterministic time complexity classes". In: *Journal of the ACM (JACM)* 25.1 (1978), pp. 146–167.

[Sha13] Igor R Shafarevich. *Basic algebraic geometry 1: Varieties in projective space*. 3rd ed. Springer Science & Business Media, 2013.

[SP08] Ilya Shpitser and Judea Pearl. "Complete identification methods for the causal hierarchy". In: *Journal of Machine Learning Research* 9.Sep (2008), pp. 1941–1979.

[Spi16] Michael Spivey. "The Chu-Vandermonde Identity via Leibniz's Identity for Derivatives". In: *The College Mathematics Journal* 47.3 (2016), pp. 219–220.

[SU22] Chandler Squires and Caroline Uhler. "Causal structure learning: A combinatorial perspective". In: *Foundations of Computational Mathematics* (2022), pp. 1–35.

[Sta00] Richard P Stanley. "Positivity problems and conjectures in algebraic combinatorics". In: *Mathematics: frontiers and perspectives* 295 (2000), p. 319.

[Sta11] Richard P. Stanley. *Enumerative Combinatorics, Volume 1*. Second. Vol. 49. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2011. ISBN: 978-1-107-60262-5.

[SZ81] Patrick Suppes and Mario Zanotti. "When are probabilistic explanations possible?" In: *Synthese* 48 (1981), pp. 191–199.

[TKO13] Balder Ten Cate, Phokion G Kolaitis, and Walied Othman. "Data exchange with arithmetic operations". In: *Proceedings of the 16th International Conference on Extending Database Technology*. 2013, pp. 537–548.

[Tod91] Seinosuke Toda. "PP is as Hard as the Polynomial-Time Hierarchy". In: *SIAM Journal on Computing* 20.5 (1991), pp. 865–877.

[Wei00] Klaus Weihrauch. *Computable Analysis - An Introduction*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000. ISBN: 978-3-540-66817-6. DOI: 10.1007/978-3-642-56999-9.

[Wei+18] Luca Weihs et al. "Determinantal generalizations of instrumental variables". In: *Journal of Causal Inference* 6.1 (2018).

[Wri28] Philip G. Wright. *Tariff on animal and vegetable oils*. Macmillan Company, New York, 1928.

[Wri21] Sewall Wright. "Correlation and causation". In: *Journal of agricultural research* 20.7 (1921), p. 557.

[Žák83] Stanislav Žák. "A Turing machine time hierarchy". In: *Theoretical Computer Science* 26.3 (1983), pp. 327–333.

[ZBL23] Benito van der Zander, Markus Bläser, and Maciej Liśkiewicz. "The Hardness of Reasoning about Probabilities and Causality". In: *Proc. Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[ZL16] Benito van der Zander and Maciej Liśkiewicz. "On Searching for Generalized Instrumental Variables." In: *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016, pp. 1214–1222.

[ZTL15] Benito van der Zander, Johannes Textor, and Maciej Liśkiewicz. "Efficiently Finding Conditional Instruments for Causal Inference". In: *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*. 2015, pp. 3243–3249.

[ZWBL22] Benito van der Zander, Marcel Wienöbst, Markus Bläser, and Maciej Liśkiewicz. "Identification in Tree-shaped Linear Structural Causal Models". In: *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2022, pp. 6770–6792.

[ZTB22] Junzhe Zhang, Jin Tian, and Elias Bareinboim. "Partial counterfactual identification from observational and experimental data". In: *Proc. International Conference on Machine Learning (ICLM)*. PMLR. 2022, pp. 26548–26558.