The Hardness of Reasoning about Probabilities and Causality

Benito van der Zander¹, Markus Bläser² and Maciej Liśkiewicz¹

¹Institute of Theoretical Computer Science, University of Lübeck, Germany
²Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

benito@tcs.uni-luebeck.de, mblaerer@cs.uni-saarland.de, liskiewi@tcs.uni-luebeck.de

Abstract

We study formal languages which are capable of fully expressing quantitative probabilistic reasoning and do-calculus reasoning for causal effects, from a computational complexity perspective. We focus on satisfiability problems whose instance formulas allow expressing many tasks in probabilistic and causal inference. The main contribution of this work is establishing the exact computational complexity of these satisfiability problems. We introduce a new natural complexity class, named suR, which can be viewed as a succinct variant of the well-studied class R, and show that these problems are complete for suR. Our results imply even stronger limitations on the use of algorithmic methods for reasoning about probabilities and causality than previous state-of-the-art results that rely only on the NP- or ∃R-completeness of the satisfiability problems for some restricted languages.

1 Introduction

Satisfiability problems play a key role in a broad range of research fields, including AI, since many real-life tasks can be reduced in a natural and efficient way to instances of satisfiability expressed in a suitable formal language. A prominent example is the Boolean satisfiability problem (SAT) whose instances represent Boolean formulas of propositional logic and ask whether there exists an assignment that satisfies a given formula. It is well known that any decision problem in the complexity class NP can be reduced in polynomial time to the SAT problem, see [Garey and Johnson, 1979].

In this work, we investigate satisfiability problems (and their validity counterparts) whose instance formulas allow expressing many tasks in probabilistic and causal inference. The formulas are specified in languages commonly used in pure probabilistic and interventional reasoning. In particular, the probabilistic language, called in Pearl’s Causal Hierarchy¹ the associational one, consists of Boolean combinations of (in)equalities involving pure probabilities as, e.g., P(y|x)

\[ \sum_{u,x,z,y} (P(u,x,z,y) - P(u)P(x|u)P(z|x)P(y|z,u))^2 = 0 \] (1)

which expresses the fact that the joint distribution can be factorized as \( P(u,x,z,y) = P(u)P(x|u)P(z|x)P(y|z,u) \), for all values \( u, x, z, y \). The causal language extends the probabilistic language by allowing additionally to use terms involving Pearl’s do-operator [Pearl, 2009] as, e.g., \( P(y|do(x)) \) which can ask hypothetical questions such as how likely would \( Y = y \) be given an intervention setting \( X \) to \( x \)? An example formula in this language is the single equality

\[ P(y|do(x)) = \sum_z P(z|x) \sum_{x'} P(y|x',z) P(x') \] (2)

which allows estimating the (total) causal effect of the intervention \( X = x \) on outcome variable \( Y = y \) via the prominent front-door adjustment [Pearl, 1995]. It is well known that this formalism enables inference of properties that are impossible to reason about from correlational data using a purely probabilistic framework [Shpitser and Pearl, 2008; Bareinboim et al., 2022].

Similarly as in the classical SAT, both languages allow natural, polynomial-time reductions of many tasks in probabilistic and causal inference to the satisfiability (or validity) problems. Prominent examples can be establishing properties of structural causal models [Glymour et al., 2014; Pearl, 2009; Koller and Friedman, 2009; Elwert, 2013] which endow researchers with graphical structures (also called causal Bayesian networks) to encode conditional independences and causal assumptions as a directed acyclic graph (DAG). For instance, the problem to decide, for a given DAG \( \mathcal{G} \) and variables \( X, Y, Z \), whether in every structural causal model compatible with the structure \( \mathcal{G} \), the causal effect of \( X \) on \( Y \) can be inferred via front-door adjustment (2) can be reduced in polynomial time to the validity problem. Indeed, for any \( \mathcal{G}, X, Y, Z \), one can compute a formula \( \varphi \) whose size is bounded by a polynomial in the number of nodes of \( \mathcal{G} \), such that the front-door adjustment is applicable, if and only if, \( \varphi \) is valid. E.g., for the DAG \( \mathcal{G} \):

¹See [Pearl and Mackenzie, 2018] for a first-time introduction to the topic.

²In our paper, by \( P(y|x), P(y|do(x)) \), etc., we mean \( P(Y=y|X=x), P(Y=y|do(X=x)) \), etc.
the formula \( \psi \) looks as follows (\( \psi_1 \land \psi_2 \) \( \Rightarrow \) \( \psi_{\text{do}} \)), where \( \psi_{\text{do}} \) is the equality (2) representing the front-door adjustment, \( \psi_1 \) is the equality (1) encoding the probability factorization, and \( \psi_2 \) is a conjunction of equalities, whose conjuncts encode edge orientations. E.g., \( X \rightarrow Z \) can be expressed as
\[
\sum_{x,z} (\mathbb{P}(x|\text{do}(z)) - \mathbb{P}(x))^2 = 0. \tag{3}
\]
The correctness of this reduction follows from the fact, that a structural causal model over \( X, Y, Z, U \) is substituted with the above DAG \( G \) if \( \psi_1 \land \psi_2 \) encodes \( G \). Interestingly, using do-calculus, one can prove that the front-door adjustment is applicable for the discussed instance which means that the formula \( \psi_1 \land \psi_2 \Rightarrow \psi_{\text{do}} \) is valid.

### 1.1 Our Contribution

Despite its importance, computational complexity aspects of the satisfiability problems for the general probabilistic and causal languages remain unexplored. In our work, we investigate these issues and establish, for the first time, the existence of a reduction of real numbers to the variables of the polynomials that satisfies the combination of (in)equalities? The example

\[
\exists x \exists y \quad x^2 - y = 0 \land x^3 - x = 0 \land (x > 0 \lor y \leq -1)
\]

illustrates a yes instance of ETR because \((x, y) = (1,1)\) is a solution of the sentence. The theory forms its own complexity class \( \mathbb{E} \mathcal{R} \) which is defined as the closure of the ETR under polynomial time many-one reductions. The importance of \( \mathbb{E} \mathcal{R} \) is underlined by the fact, that many meaningful problems including algebraic, geometric, graph- and game-theory, machine learning, and continuous constraint satisfaction problems, have been classified as complete for \( \mathbb{E} \mathcal{R} \), see e.g. [Schaefer and Štefankovič, 2017; Abrahamsen et al., 2018; Garg et al., 2018; Abrahamsen et al., 2021; Miltzow and Schmiermann, 2022]. Moreover, assuming \( \mathbb{N} \neq \mathbb{E} \mathcal{R} \) which is widely believed, the \( \mathbb{E} \mathcal{R} \)-completeness of a problem reflects the limitation on the use of algorithmic approaches to solve the problem, as, e.g., dynamic programming, divide and conquer, tree-width based algorithms, or SAT-solver approaches which are applicable for instances of \( \mathbb{N} \mathcal{P} \)-complete problems. The \( \mathbb{E} \mathcal{R} \)-completeness of \( \mathbb{S} \mathcal{P} \mathcal{R} \) and \( \mathbb{S} \mathcal{A} \mathcal{T} \) (Theorem 1.1) imply even stronger algorithmic limitations than could be deduced from the \( \mathbb{E} \mathcal{R} \)-completeness results of [Mossé et al., 2022] (discussed below) which concerns, however, some restricted variants of the languages.

### 1.3 Previous Work on the Hardness of Satisfiability Problems

In the past decades, several research groups have studied the computational complexity aspects of satisfiability problems for some limited languages of probabilistic and causal inference. In their pioneering work, Fagin et al. [1990] consider a language for reasoning about pure probabilities, which consists of Boolean combinations of linear (in)equalities over probabilities, like \( \mathbb{P}(X = 1 \lor X = 2) + 2\mathbb{P}(Y = 1) \leq 1/3 \land \mathbb{P}(Y = 1) \geq 2/3 \). The authors provide a complete axiomatization for the logic and show that the problem of deciding satisfiability is \( \mathbb{N} \mathcal{P} \)-complete which, surprisingly, is no worse than that of propositional logic. Later on, further studies explored the complexity aspects of probability logics [Abadi and Halpern, 1994; Speranski, 2013] and reasoning about causal models [Halpern, 2000; Eiter and Lukasiewicz, 2002; Aleksandrowicz et al., 2017].

Most germane to our work are the recent studies of Ibeling and Icard [2020] and Mossé et al. [2022] which extended the formalism of [Fagin et al., 1990], providing the more expressive languages for the second (interventional) and third (counterfactual) level of Pearl’s Causal Hierarchy. The languages consist of Boolean combinations of polynomial (in)equalities over pure probabilities or over probabilities involving the do-operator, respectively. In particular, they allow to express conditioning, as, e.g., \( \mathbb{P}(y|x) = \mathbb{P}(y|x)/\mathbb{P}(x) \).

But the languages limit the use of marginalization since the summation operator \( \Sigma \) over the domain of random variables, as used, e.g., in Eq. (1)-(3), is not allowed. Consequently, to express the marginal distribution of a variable \( Y \) over a subset of variables \( \{Z_1, \ldots, Z_m\} \subseteq \{X_1, \ldots, X_n\} \), as \( \sum_{z_1, \ldots, z_m} \mathbb{P}(y, z_1, \ldots, z_m) \), in the language without summation requires an extension \( \mathbb{P}(y, Z_1 = 0, \ldots, Z_m = 0) + \ldots + \mathbb{P}(y, Z_1 = 1, \ldots, Z_m = 1) \) of exponential size in \( m \). (In the expression, we assume that the \( Z_i \) are binary variables). Similarly, \( \mathbb{P}(y|\text{do}(x)) = \sum\mathbb{P}(y|x, z)\mathbb{P}(z) \) is expressed equivalently as \( \mathbb{P}(y|\text{do}(x)) = \mathbb{P}(y|x, Z=0)\mathbb{P}(Z=0) + \)
\( P(y|x, Z=1)P(Z=1) \). While this might be acceptable for one variable \( Z \), the expansion of the sums grows exponentially when we have several variables. Therefore, having an explicit summation operator is highly desirable.

Ibeling and Icard [2020] and Mossé et al. [2022] give complete axiomatizations of these languages and investigate the computational complexity of the satisfiability problem for each language. In particular, they prove that, although the languages for interventional and counterfactual inference are more expressive than the ones for probabilistic reasoning, the computational complexities of the satisfiability problem for the logics are both \( \#P \)-complete. Combining these results with ours, we obtain a precise complexity theoretic classification of the satisfiability problems for all three levels of Pearl’s Causal Hierarchy combined with various languages for terms involving probabilities (proven by 1: Fagin et al. [1990], 1: Mossé et al. [2022]):

<table>
<thead>
<tr>
<th>Terms</th>
<th>prob.</th>
<th>interven.</th>
<th>counterf.</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>lin</td>
<td>NP-complete</td>
<td>†, ‡</td>
<td>†</td>
<td></td>
</tr>
<tr>
<td>poly</td>
<td>#P-complete</td>
<td>†</td>
<td></td>
<td></td>
</tr>
<tr>
<td>poly &amp; \Sigma</td>
<td>succ #P-complete</td>
<td>succ #P-hard</td>
<td>Thm. 1.1</td>
<td></td>
</tr>
</tbody>
</table>

While the languages of [Mossé et al., 2022] are capable of fully expressing quantitative probabilistic reasoning, respectively, do-calculus reasoning for causal effects, as discussed above, due to the lack of the summation operator \( \Sigma \), they do not capture the standard notation used commonly in probabilistic and causal inference. Consequently, to analyze the computational complexity aspects of the inference, languages that exploit the standard notation need to be used. Indeed, for the computational complexity, the key issue is how the instances are represented since the time or space complexities are functions in the length of the input. For example, if the language as in [Mossé et al., 2022] would be used (instead of the succinct one with summation operator), then the time complexity of many algorithms, including, e.g., the seminal Shpitser and Pearl [2006] algorithm to estimate the interventional distribution, would jump from polynomial to exponential. Another problem would also be to construct polynomial time reductions to the SAT problems since using the succinct encodings as, e.g., in Eq. (1)–(3), would not be allowed.

**Structure of this Paper.** The remainder of this work is dedicated to proving Theorem 1.1. After providing preliminaries (Sec. 2 and 3), we introduce the class \( \text{succ} \#P \) in Sec. 4 and provide the first complete problems for \( \text{succ} \#P \) in Sec. 5. In Sec. 6, we prove the membership of \( \text{SAT}_{\text{prob}} \) and \( \text{SAT}_{\text{interven}} \) in the class \( \text{succ} \#P \) and their \( \text{succ} \#P \)-hardness. The omitted proofs can be found in the appendix of a companion paper [van der Zander et al., 2023].

# 2 Preliminaries

## 2.1 Syntax of Probabilistic and Causal Languages

The syntax used in this paper extends the definitions in Ibeling and Icard, 2020; Mossé et al., 2022 for the languages of the first two levels of Pearl’s Causal Hierarchy. We consider discrete distributions and represent the values of the random variables as \( \text{Val} = \{0, 1, \ldots, c - 1\} \) and denote by \( X \) the set of all (endogenous) random variables. By capital letters \( X_1, X_2, \ldots \), we denote the individual variables and assume, w.l.o.g., that they all share the same domain \( \text{Val} \). A value of \( X_i \) is denoted by \( x_i \) or a natural number.

In order to reason about the (in)equality of arithmetic terms involving probability expressions, we need to define languages describing probabilities, languages describing arithmetic terms, and languages describing (in)equalities. The first languages characterize probabilistic and causal events as Boolean conditions over the values of (endogenous) random variables:

\[
\begin{align*}
\mathcal{L}_{\text{prop}} & := X = x \mid \neg \mathcal{L}_{\text{prop}} \mid \mathcal{L}_{\text{prop}} \land \mathcal{L}_{\text{prop}} \\
\mathcal{L}_{\text{interven}} & := \top \mid X = x \mid \mathcal{L}_{\text{int}} \land \mathcal{L}_{\text{int}} \\
\mathcal{L}_{\text{post-int}} & := [\mathcal{L}_{\text{int}}] \mathcal{L}_{\text{prop}}
\end{align*}
\]

where \( X \in X \), and \( x \) is either in \( \text{Val} \) or is a summation variable as defined below.

To make the notation more convenient for our analysis, we use \( [\mathcal{L}_{\text{int}}] \) in the syntax above to describe an intervention on certain variables. The operator \( [\mathcal{L}] \delta \) creates a new post-intervention model and the assigned propositional formula \( \delta \) only applies to the new model. Thus, for example, \( \mathbb{P}(X=x|Y=y) \), which, using do-notation can be expressed as \( \mathbb{P}(Y=y| \text{do}(X=x)) \), denotes the probability that the variable \( Y \) equals \( y \) in the model after the intervention \( \text{do}(X=x) \). Since \( \top \) means that no intervention has been applied, we can assume that \( \mathcal{L}_{\text{prop}} \subseteq \mathcal{L}_{\text{post-int}} \).

Inserting the primitives into arithmetic expressions, we get the language \( T(L) \), where, for \( L \in \{ \mathcal{L}_{\text{prop}}, \mathcal{L}_{\text{post-int}} \} \), \( \delta \in L \), and \( \delta' \in L_{\text{prop}} \), any \( e \in T(L) \) is formed by the grammar:

\[
e ::= \mathbb{P}(\delta|\delta') | e + e' | e \cdot e' | \sum_x e.
\]

(5)

The probabilities of the form \( \mathbb{P}(\delta) \) or \( \mathbb{P}(\delta|\delta') \), are called primitives or (atomic) terms.

In the summation operator \( \sum_x \) in definition (5), we have a dummy variable \( x \) which ranges over all values \( 0, 1, \ldots, c - 1 \). The summation \( \sum_x e \) is a purely syntactical concept which represents the sum \( \mathbb{E}[e|\delta] = \mathbb{E}[e_1|\delta_1] + \ldots + \mathbb{E}[e_c|\delta_c] \), where, by \( \mathbb{E}[e|\delta] \), we mean the expression in which all occurrences of \( x \) are replaced with value \( v \). E.g., for \( \text{Val} = \{0, 1\} \), 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.

Finally, we define the languages of Boolean combinations of inequalities, following the grammar, where \( e, e' \) are expressions in \( T(L_{\text{prop}}) \) for \( \text{prob} \) and \( T(L_{\text{post-int}}) \) for \( \text{causal} \), respectively:

\[
\begin{align*}
\mathcal{L}_{\text{prob}} & := e \leq e' \mid \neg \mathcal{L}_{\text{prob}} \mid \mathcal{L}_{\text{prob}} \land \mathcal{L}_{\text{prob}} \\
\mathcal{L}_{\text{interven}} & := e \leq e' \mid \neg \mathcal{L}_{\text{interven}} \mid \mathcal{L}_{\text{interven}} \land \mathcal{L}_{\text{interven}}
\end{align*}
\]

Together with conjunctions, the comparisons yield an equality relation. Moreover, for the atomic formula \( X=x \) in

\( \quad ^3\)This is a recursive definition where, e.g., \( e + e' \) means, for \( e, e' \in T(L) \), the expression \( e + e' \) is also in \( T(L) \).

\( \quad ^4\)As usually, \( \mathbb{P}(Y=1, X=x) \), etc., means \( \mathbb{P}(Y=1 \land X=x) \), etc.
probabilities as, e.g. $P(X=x)$, we use a common abbreviation $P(x)$ if this does not lead to confusion. Thus, for example, Eq. (1) builds the correct formula in the language $L_{\text{prob}}$ and Eq. (2) and (3) are the correct formulas in $L_{\text{interven}}$.

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., $P(x) = P(y)$ means $P(x) \geq P(y) \land P(y) \geq P(x)$. Subtraction or divisions can be encoded by moving a term to the other side of the equation, e.g., $P(x) - P(y) = P(z)$ means $P(x) = P(z) + P(y)$, and $P(x)/P(y) = P(z)$ means $P(x) = P(z)P(y)$. Any positive integer can be encoded from the fact that variables $\alpha$ can be encoded by moving a term to the other side of the equation, e.g., $4 \equiv (1+1)(1+1) \equiv (P(T) + P(T))(P(T) + P(T))$. The number 0 can be encoded as inconsistent probability, i.e., $P(\bar{x}) \land \bar{x} = 2$.

We now define succinct encodings of instances for $\text{ETR}$ and $\text{SAT}_{\text{interven}}$ and $\text{SAT}_{\text{prob}}$, respectively, and asks whether there exists a model $M$ such that $M \models \varphi$. Analogously, we define the validity problems for languages $\text{ETR}_{\text{interven}}$ and $\text{ETR}_{\text{prob}}$ of deciding whether, for a given $\varphi$, $M \models \varphi$ holds for all models $M$. From the definitions, it is obvious that causal variants of the problems are at least as hard as the prob counterparts.

To give a first intuition about the expressive power of the $\text{ETR}_{\text{prob}}$ problem, we prove the following result which shows that $\text{ETR}_{\text{prob}}$ can encode any problem in $\text{NEXPTIME}$ efficiently.

**Proposition 3.1.** The $\text{ETR}_{\text{prob}}$ problem is $\text{NEXPTIME}$-hard.

The remaining part of the paper is devoted to proving the main result (Theorem 1.1) showing that the satisfiability problems are complete for the new class $\text{co-Succ} \text{ETR}$. Since a formula $\varphi$ is not valid, if and only if $\neg \varphi$ is valid and $\neg \varphi$ is in $\text{ETR}_{\text{prob}}$ or in $\text{ETR}_{\text{interven}}$, respectively, we can conclude from Theorem 1.1:

**Corollary 3.2.** The validity problems for the languages $\text{ETR}_{\text{prob}}$ and $\text{ETR}_{\text{interven}}$ are complete for $\text{co-Succ} \text{ETR}$, which is related to standard classes as follows: $\text{co-NEXPTIME} \subseteq \text{co-SuccETR} \subseteq \text{EXPSPACE}$.

### 4 The Existential Theory of the Reals, Succinctly

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

$$\exists x_1 \ldots \exists x_n \varphi(x_1, \ldots, x_n).$$

$\varphi$ is a quantifier-free Boolean formula over the basis $\{\vee, \land, \neg\}$ 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.

All operations have an arity of at most two, so we can represent $\varphi$ as a binary tree. The leaves of the tree are labeled with variables or constants, the inner nodes are labeled with $+$, $\cdot$, $\leq$, $<$, $\leq$, $\land$, $\lor$, and $\neg$. $\neg$-nodes have in-degree one, and all other inner nodes have in-degree two.

We now define succinct encodings of instances for ETR. Succinct encodings have been studied for problems in $\text{NP}$, see e.g. [Papadimitriou, 1994]. The input is now a Boolean circuit $C$ computing a function $\{0, 1\}^N \rightarrow \{0, 1\}^M$, which encodes the input instance. For example, an instance of the well-known 3-SAT problem is some standard encoding of a given 3-CNF formula. In the succinct version of 3-SAT, we are given a circuit $C$. $C$ encodes a 3-CNF formula $\psi$ in the following way: $C(i)$ is an encoding of the $i$th clause of $\psi$. In
this way, one can encode an exponentially large 3-SAT formula $\psi$ (with up to $2^n$ clauses). The succinct encoding typically induces a complexity jump: The succinct version of the 3-SAT problem is NP-complete. The same is true for many other NP-complete problems [Papadimitriou, 1994].

In the case of ETR, the formula $\psi$ is now given succinctly by a Boolean circuit $C$. The circuit $C$ computes a function $\{0,1\}^N \rightarrow \{0,1\}^M$. The input is (an encoding of) a node $v$ of the formula and the output $C(v)$ contains the label of the node as well as its parent and its children (encoded as a string in binary). In this way, we can represent a formula that is exponentially large in the size of the circuit.

**Example 4.1.** Consider the formula $\exists x_1 : x_1^2 = 1 + 1$ (“2 has a square root over the reals”). As an instance of ETR, this formula would be encoded as a binary string in some standard way. Figure 1 shows the binary tree underlying the formula. There are seven nodes, which we can represent as integers $1, \ldots, 7$. As a succinct ETR instance, the formula will be given by a circuit $C$, computing a function $\{1, \ldots, 7\} \rightarrow \{=, +, 0, 1, x_1\} \times \{0, 1, \ldots, 7\}$ describing the tree locally. For instance, $C(1) = (=, 0, 2, 3)$, means the following: the label of the node 1 is $=$. It has no parent, that is, it is the root, which is indicated by giving 0 as the parent. The left child is 2 and the right child is 3. In the same way $C(4) = (x_1, 2, 0, 0)$, which means that the node 4 is a leaf with parent 2 and it is labeled with the variable $x_1$. We can represent the integers $\leq 7$ as bit strings of length $N = 3$ and the tuples $C(i)$, e.g., $=, 0, 2, 3$ or $x_1, 2, 0, 0$, can be encoded as bit strings of some length $M$. Thus, $C$ can be considered a function $C : \{0,1\}^N \rightarrow \{0,1\}^M$. Note that we cannot use more variables than the formula has nodes, thus to encode the tuple $C(i)$, we need at most $M \leq (3 + N) + 3N = 4N$ bits: in the first entry, we need $N + 1$ bits to encode a label $+, , =, <, \leq, \land, \lor, \varnothing$ or an index $j$ of a variable $x_j$; to encode each of the remaining components, $N$ bits suffice. It is clear that one can construct such a circuit $C$. When the formula is large but also structured, then the circuit $C$ can be much smaller than the formula itself.

**Definition 4.2 (succETR).** succETR is the set of all Boolean circuits $C$ that encode a true sentence $\varphi$ as in (6) as follows. Assume that $C$ computes a function $\{0,1\}^N \rightarrow \{0,1\}^M$. Then $\{0,1\}^N$ is the node set of the tree underlying $\varphi$ and $C(i)$ is the encoding of the description of node $i$, consisting of the label of $i$, its parent, and its two children. The variables in the formula are $x_1, \ldots, x_N$.

The number of variables is formally always a power of 2. If we need fewer variables, we simply do not use them in the formula. We still quantify over all of them, but this is no problem. In the same way, the number of nodes is always a power of 2. If we need fewer nodes, we can define a special value for $C(i)$, which marks node $i$ as unused.

We will always assume that all negations are pushed to the lowest level of the formula and then they are absorbed in the arithmetic terms, i.e., $\neg(s = t)$ becomes $(s < t) \lor (t < s)$, $\neg(s < t)$ becomes $t \leq s$ and $\neg(s \leq t)$ becomes $t < s$. Furthermore, we can remove the comparisons $\leq$, since we can replace $s \leq t$ by $(s < t) \lor (s = t)$.

**Remark 4.3.** In the non-succinct case, eliminating the negations by pushing them to the bottom is no problem, as it can be done explicitly. In the succinct case, it seems that we need to assume that already the given input has this property.

Finally, based on the succETR problem, we define the succinct version of $\exists R$ as follows:

**Definition 4.4.** succ$\exists R$ is the class of all problems that are polynomial time reducible to succETR.

succ$\exists R$ is related to known complexity classes as follows.

**Theorem 4.5.** $\text{NEXPTIME} \subseteq \text{succ} \exists R \subseteq \text{EXPSPACE}$.

### 5 First Complete Problems for succ$\exists R$

In this section, we present the first problems which are complete for succ$\exists R$. The key role plays the problem $\Sigma_q$-ETR which we need to prove the membership of $\text{SAT}_{\text{Interven}}$ in the class succ$\exists R$ in Proposition 6.1.

QUAD is the problem to decide whether a given family of quadratic multivariate polynomials has a common root. Here the polynomials are given as lists of monomials. The coefficient is represented as a quotient of two integers. QUAD is a complete problem for the class $\exists R$ of all problems that are reducible to ETR [Schaefer and Stefaňkovič, 2017].

The succinct version succQUAD is defined as follows: We get a Boolean circuit $C$ as input computing a function $\{0,1\}^K \times \{0,1\}^N \rightarrow \{0,1\}^M$. $C(x,y)$ is an encoding of the $y$th monomial of the $x$th equation.

We allow that a monomial appears several times in these lists and the coefficients all add up, so the final polynomials are the sum of all monomials in the list.

**Lemma 5.1.** succETR $\leq_P$ succQUAD. Furthermore, the reduction only creates quadratic polynomials with at most four monomials and coefficients $\pm 1$.

$\Sigma_q$-ETR is defined like ETR, but we add to the signature an additional summation operator. This is a unary operator $\sum_{x=a}^b$. Consider an arithmetic term given by a tree with the top gate $\sum_{x=a}^b$. Let $(x_1, \ldots, x_n)$ be the term computed at the child of the top gate. Then the new term computes $\sum_{x=a}^b (x_1, \ldots, x_n)$. That is, we replace the variable $x_j$ by a summation variable $e$, which then runs from $a$ to $b$.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sum_{x=a}^b$</td>
<td>Sums the variables $x_j, \ldots, x_n$ over the interval $[a, b]$</td>
</tr>
<tr>
<td>$\sum_{x=a}^b (x)$</td>
<td>Sums the variables $x_j, \ldots, x_n$ over the interval $[a, b]$</td>
</tr>
<tr>
<td>$\sum_{x=a}^b (e)$</td>
<td>Sums the variables $x_j, \ldots, x_n$ over the interval $[a, b]$</td>
</tr>
</tbody>
</table>

Furthermore, we allow a new form of variable indexing in $\Sigma_q$-ETR, namely variables of the form $x_n(x_j, \ldots, x_m)$. This can only be used when variables $x_{j_1}, \ldots, x_{j_m}$ occur in the scope of summation operators and are replaced by summation variables $e_{j_1}, \ldots, e_{j_m}$ with summation range $\{0,1\}^m$.  


Page dimensions: 612.0x792.0

Theorem 5.3. \( \Sigma_{\nu}^{\text{ETR}} \) is \( \Sigma_{\nu}^{\text{ETR}} \) complete.

Lemma 6.3. \( \Sigma_{\nu}^{\text{ETR}} \) complete.

Lemma 5.2. \text{succQUAD} \leq_p \Sigma_{\nu}^{\text{ETR}} \leq_p \text{succETR} \quad \text{which yields:}

Theorem 5.3. The problems \text{succQUAD} and \( \Sigma_{\nu}^{\text{ETR}} \) are \text{succETR} complete.

6 Proof of the Main Theorem

Equipped with the tools provided in the previous sections, we are ready to prove our main result (Theorem 1.1) by showing:

\[
\text{SAT}_{\text{prob}}, \text{SAT}_{\text{interven}} \in \text{succETR} \quad \text{and} \quad \text{SAT}_{\text{prob}}, \text{SAT}_{\text{interven}} \text{ are succETR-hard.} \quad (7)
\]

We first show the membership of \text{SAT}_{\text{prob}} in the class \text{succETR} proving the following reduction:

Proposition 6.1. \( \text{SAT}_{\text{interven}} \leq_p \Sigma_{\nu}^{\text{ETR}} \).

Due to the transitivity of \( \leq_p \) and by Theorem 5.3, we get that \( \Sigma_{\nu}^{\text{ETR}} \leq_p \text{succETR} \). Moreover, since \( \text{SAT}_{\text{prob}} \leq_p \Sigma_{\nu}^{\text{ETR}} \quad \text{(7)} \) follows.

For (8), it is sufficient to show the \text{succETR}-hardness of \( \text{SAT}_{\text{prob}} \). We prove this in Proposition 6.5. To this aim, we introduce the problem \( \text{succETR}^{1/s+1/x}_{[-1/s,1/s]} \) and prove the \text{succETR}-hardness of this problem.

Definition 6.2 (Abrahamsen et al. 2017). In the problem \( \text{ETR}^{5+1, x} \), where \( c \in \mathbb{R} \), we are given a set of real variables \( \{x_1, \ldots, x_n\} \) and a set of equations of the form \( x_i = c \), \( x_i + x_j = x_k \), \( x_i \cdot x_j = x_k \), for \( i, j, k \in [n] \). The goal is to decide whether the system of equations has a solution. The problem, where we also require that \( x_1, \ldots, x_n \in [a,b] \), for some \( a, b \in \mathbb{R} \), is denoted by \( \text{ETR}^{5+1, x}_{[a,b]} \).

Lemma 6.3 (Abrahamsen et al. 2017). \( \text{ETR}^{5+1, x} \) and \( \text{ETR}^{5+1, x}_{[-1/s,1/s]} \) are \( \Sigma_{\nu}^{\text{ETR}} \) complete.

Let \( \text{succETR}^{5+1, x} \) and \( \text{succETR}^{5+1, x}_{[a,b]} \) denote the succinct versions of \( \text{ETR}^{5+1, x} \) and \( \text{ETR}^{5+1, x}_{[a,b]} \), respectively. We assume that their instances are represented by seven Boolean circuits \( C_0, C_1, \ldots, C_6 : \{0,1\}^M \rightarrow \{0,1\}^N \) such that \( C_0(j) \) gives the index of the variables in the \( j \)-th equation of type \( x_i = 1/s \), \( C_1(j), C_2(j), C_3(j) \) give the indices of variables in the \( j \)-th equation of the type \( x_i + x_j = x_k \), and \( C_4(j), C_5(j), C_6(j) \) give the indices of variables in the \( j \)-th equation of the type \( x_i \cdot x_j = x_k \). Without loss of generality, we can assume that an instance has the same number \( 2^M \) of equations of each type; if not, one of the equations can be duplicated as many times as needed.

Lemma 6.4. \( \text{succETR}^{5+1, x}_{[-1/s,1/s]} \) is \( \text{succETR} \)-complete.

To establish the lemma, we first show that \text{succQUAD} \( \leq_p \) \( \text{succETR}^{5+1, x} \) and then prove that \( \text{succETR}^{5+1, x} \leq_p \) \( \text{succETR}^{5+1, x}_{[-1/s,1/s]} \) (see van der Zander et al., 2023).

Now we are ready to prove the \text{succETR}-hardness of \text{SAT}_{\text{prob}} through the following reduction:

Proposition 6.5. \( \text{succETR}^{5+1, x}_{[-1/s,1/s]} \leq_p \text{SAT}_{\text{prob}} \).

Proof. Let us assume that the instance of \( \text{succETR}^{5+1, x}_{[-1/s,1/s]} \) is represented by seven Boolean circuits \( C_0, C_1, \ldots, C_6 : \{0,1\}^M \rightarrow \{0,1\}^N \) as described above. Let the variables of the instance be indexed as \( x_{e_1}, \ldots, x_{e_N} \), with \( e_i \in \{0,1\} \) for \( i \in [N] \). Below, we often identify the bit sequence \( b_1, \ldots, b_L \) by an integer \( j \), with \( 0 \leq j \leq 2^L - 1 \), the binary representation of which is \( b_1, \ldots, b_L \) and vice versa.

The instance of the problem \( \text{succETR}^{5+1, x}_{[-1/s,1/s]} \) is satisfiable if and only if:

\[
\exists x_{0}, x_{2N+1} \in [-1/s,1/s] \quad \sum_{j=0}^{2^{M}-1} \left( (x_{C_0(j)} - 1/s)^2 + x_{C_1(j)} + x_{C_2(j)} - x_{C_3(j)} \right)^2 + \left( x_{C_4(j)} \cdot x_{C_5(j)} - x_{C_6(j)} \right)^2 = 0. \quad (9)
\]

We construct a system of equations in the language \( \text{L}_{\text{prob}} \) and prove that there exists a model which satisfies the equations if and only if the formula (9) is satisfiable.

Let \( X_0, X_1, \ldots, X_N \) be binary random variables. We will model each real variable \( x_{e_1}, \ldots, x_{e_N} \) as a term involving the conditional probability \( \mathbb{P}(X_0=0 \mid X_1=e_1, \ldots, X_N=e_N) \) as follows:

\[
q_{e_1, \ldots, e_N} := \frac{2}{3} \cdot \mathbb{P}(X_0=0 \mid X_1=e_1, \ldots, X_N=e_N) - \frac{1}{3}.
\]

This guarantees that \( q_{e_1, \ldots, e_N} \in [-1/s,1/s] \). In our construction, the existential quantifiers in the formula (9) correspond to the existence of a probability distribution \( \mathbb{P}(X_0, X_1, \ldots, X_N) \) which determines the values \( q_{e_1, \ldots, e_N} \). This means that the formula (9) is satisfiable if and only if there exists a model for the equation:

\[
\sum_{j=0}^{2^{M}-1} \left( (q_{C_0(j)} - 1/s)^2 + (q_{C_1(j)} + q_{C_2(j)} - q_{C_3(j)})^2 + (q_{C_4(j)} \cdot q_{C_5(j)} - q_{C_6(j)})^2 \right) = 0. \quad (10)
\]

The challenging task which remains to be solved is to express the terms \( q_{C_i(j)} \) in the language \( \text{L}_{\text{prob}} \). Below we provide a system of equations that achieves this goal.

To model a Boolean formula encoded by a node of \( C_i \), with \( i = 0,1, \ldots, 6 \), we use arithmetization to go from logical formulas to polynomials over terms involving probabilities of the form \( \mathbb{P}(\delta) \). We start with input nodes \( v \in \{0,1\} \) and model them as events \( \delta_v \) such that \( \mathbb{P}(\delta_v) = 1 \) if and only if \( v = 1 \). For every internal node \( v \) of \( C_i \), we proceed as follows. If \( v \) is labeled with \( \neg \) and \( u \) is a child of \( v \), then we define an event \( \delta_u \) such that \( \mathbb{P}(\delta_u) = 1 - \mathbb{P}(\delta_v) \). If \( v \) is labeled with \( \land \) and \( u \) and \( w \) are children of \( v \), then we specify an event \( \delta_u \) such that \( \mathbb{P}(\delta_u) = \mathbb{P}(\delta_v) \mathbb{P}(\delta_w). \) Finally, if \( v \) is labeled with \( \lor \) and \( u \) and \( w \) are children of \( v \), then \( \mathbb{P}(\delta_v) = 1 - (1 - \mathbb{P}(\delta_u))(1 - \mathbb{P}(\delta_w)) \). Thus, if \( v \) is an output node of a circuit \( C_i \), then, for \( C_i \) fed with input \( j = b_1 \ldots b_M \in \{0,1\}^M \), we have \( v = 1 \) if and only if \( \mathbb{P}(\delta_v) = 1 \). We define the events as follows.
Let $u_{i,1}, \ldots, u_{i,M}$ be the input nodes of $C_i$ and let, for every node $v$ of $C_i$, the sequence $\ell_{v,1}, \ldots, \ell_{v,L}$ denotes the indices of the leaves $u_{i,\ell_{v,1}}, \ldots, u_{i,\ell_{v,L}}$ of the sub-circuit with root $v$. Note that, for an input $u_{i,k}$, the index $\ell_{u_{i,k}} = k$ and $w_{u_{i,k}} = 1$. To define the events, for every node $v$, we introduce a sequence of $M$ new binary random variables $X_{v,1}, \ldots, X_{v,M}$ and provide an equation that enforces the properties described above. The probabilities used in the equations involve, for a node $v$, only the variables indexed with $\ell_{v,1}, \ldots, \ell_{v,L}$; the remaining variables are irrelevant.

For the $k$th input node $u_{i,k}$ and the variable $X_{u_{i,k},k}$, we require

$$\mathbb{P}(X_{u_{i,k},k}=0) = 0 \text{ and } \mathbb{P}(X_{u_{i,k},k}=1) = 1. \quad (11)$$

For every internal node $v$ of $C_i$, if $v$ is labeled with $\neg$ and $u$ is a child of $v$, then we define the equation:

$$\sum_{b_1, \ldots, b_{k_v}} (\mathbb{P}(X_{v,\ell_{v}}=b_1, \ldots, X_{v,\ell_{v}}=b_{k_v})) - (1 - \mathbb{P}(X_{v,\ell_{v}}=b_1, \ldots, X_{v,\ell_{v}}=b_{k_v}))^2 = 0. \quad (12)$$

If $v$ has label $\land$ and $w$ and $z$ are children of $v$, then we define

$$\sum_{b_1, \ldots, b_{k_v}} (\mathbb{P}(X_{v,\ell_{v}}=b_1, \ldots, X_{v,\ell_{v}}=b_{k_v}) - \mathbb{P}(X_{w,\ell_{w}}=b_1, \ldots, X_{w,\ell_{w}}=b_{k_w}) \times \mathbb{P}(X_{z,\ell_{z}}=b_1, \ldots, X_{z,\ell_{z}}=b_{k_z}))^2 = 0, \quad (13)$$

where the indices $j_k, j'_k \in \{1, \ldots, k_v\}$ are defined as $j_k := i$ for $\ell_{v}^k = \ell_{w}^k$ and $j'_k := i$ for $\ell_{v}^k = \ell_{z}^k$.

If $v$ is labeled with $\lor$, we encode it as a negation of $\land$ after negating the children of $v$.

Let $\bar{u}_{i,1}, \ldots, \bar{u}_{i,N}$ be the output nodes of $C_i$. For the $k$th output $v = \bar{u}_{i,k}$ and for an $C_i$’s input $j = b_1 \ldots b_M$, we denote by $\alpha_{i,k}(j)$ the expression

$$\alpha_{i,k}(j) := \mathbb{P}(X_{v,\ell_{v}}=b_1, \ldots, X_{v,\ell_{v}}=b_{k_v})$$

which represents the value of the $k$th output bit of $C_i(j)$ in such a way that the bit is equal to $e_k$ if and only if the probability $\alpha_{i,k}(j) = e_k$. We illustrate this concept in Example 6.6.

Finally, for $i = 0, 1, \ldots, 6$ and $j = b_1 \ldots b_M \in \{0, 1\}^M$, let

$$\alpha(i, j) := \sum_{e_0, \ldots, e_N \in \{0, 1\}} q_{e_0, \ldots, e_N} \times \prod_{k=1}^{N} (\alpha_{i,k}(j) \mathbb{P}(E=e_k) + (1 - \alpha_{i,k}(j))(1 - \mathbb{P}(E=e_k))),$$

where $E$ is a new binary random variable with $\mathbb{P}(E=0) = 0$ and $\mathbb{P}(E=1) = 1$. The crucial property of this encoding is that: For all $i = 0, 1, \ldots, 6$ and $j = b_1 \ldots b_M \in \{0, 1\}^M$, it is true that $\alpha(i, j) = q_{C_i(j)}$.

Now, replace every term $q_{C_i(j)}$ in the Eq. (10) by $\alpha(i, j)$ and get the following final equation in the language $L_{prob}^*$:

$$\sum_{j=0}^{2^M-1} \left( (\alpha(0, j) - 1/8)^2 + (\alpha(1, j) + \alpha(2, j) - \alpha(3, j))^2 + (\alpha(4, j) \cdot \alpha(5, j) - \alpha(6, j))^2 \right) = 0. \quad (14)$$

This completes the proof since it is true that the formula (9) is satisfiable if and only if there exists a model which satisfies Eq. (11)–(13) and Eq. (14). Obviously, the size of the resulting system of equations is polynomial in the size $|C_0| + |C_1| + \ldots + |C_6|$ of the input instance and the system can be computed in polynomial time.

Example 6.6. Consider a Boolean circuit $C$ with three input nodes $u_1, u_2, u_3$, internal nodes $v, w$, and one output node $\bar{u}$ as shown in Fig. 2. The relevant variables used to encode the output bit are: $X_{u_1,1}, X_{u_2,2}, X_{u_3,3}$, assigned to the input nodes, as well as $X_{v,1}, X_{v,2}$ and $X_{w,2}, X_{w,3}$ and $X_{\bar{u}_1}, X_{\bar{u}_2}, X_{\bar{u}_3}$, assigned to $v$, $w$, and $\bar{u}$, respectively. Then, assuming the constraints expressed in Eq. (11)–(13) are satisfied, for any $b_1, b_2, b_3 \in \{0, 1\}$, the probability $o(j = b_1 b_2 b_3) := \mathbb{P}(X_{\bar{u}_1}=b_1, X_{\bar{u}_2}=b_2, X_{\bar{u}_3}=b_3)$ can be estimated as follows:

$$o(j) = \mathbb{P}(X_{\bar{u}_1}=b_1, X_{\bar{u}_2}=b_2, X_{\bar{u}_3}=b_3)$$

$$= \mathbb{P}(X_{u_1}=b_1, X_{v,2}=b_2) \cdot \mathbb{P}(X_{w,2}=b_2, X_{w,3}=b_3)$$

$$= \mathbb{P}(X_{u_1}=b_1) \cdot \mathbb{P}(X_{u_2}=b_2) \cdot \mathbb{P}(X_{u_3}=b_3) \cdot (1 - (1 - \mathbb{P}(X_{u_2}=b_2)) \cdot (1 - \mathbb{P}(X_{u_3}=b_3)))$$

$$= b_1 \cdot b_2 \cdot (1 - (1 - b_2) \cdot (1 - b_3)),$$

which is 0 if $C(b_1, b_2, b_3) = (b_1 \land b_2) \land (b_2 \lor b_3) = 0$, and it is equal to 1 if $C(b_1, b_2, b_3) = 1$.

7 Discussion

We have analyzed the complexity of deciding whether a system of (in)equalities involving probabilistic and causal formulas is satisfiable using standard languages, allowing formulas with summations. We have shown that the problems are complete for a new complexity class succ\existsR. Using these results, we could conclude that the complexity of the validity problem, asking whether a Boolean combination of (in)equalities involving pure probabilities and interventional expressions is valid, is complete for the class co-succ\existsR.

Any \existsR-complete problem can probably be turned into a succ\existsR-complete one by encoding the input succinctly. Although we do not know other published succ\existsR-complete problems, the situation is similar to the relation of \NP and NEXPTIME. The succinct versions of almost all \NP-complete problems are NEXPTIME-complete. We think that succ\existsR is so natural that further complete problems not coming from \existsR-complete ones will be found in the future.
Acknowledgments
This work was supported by the Deutsche Forschungsgemeinschaft (DFG) grant 471183316 (ZA 1244/1-1).

Contribution Statement
Markus Bläser and Maciej Liśkiewicz are equally last authors.

References


