The Complexity of Satisfiability in Non-Iterated and Iterated Probabilistic Logics

Ioannis Kokkinis

Ioannis Kokkinis
Technical University of Dortmund
ioannis.kokkinis@tu-dortmund.de

December 5, 2017

Abstract

Let $L$ be some extension of classical propositional logic. The non-iterated probabilistic logic over $L$, is the logic PL that is defined by adding non-nested probabilistic operators in the language of $L$. For example in PL we can express a statement like “the probability of truthfulness of $A$ is at 0.3” where $A$ is a formula of $L$. The iterated probabilistic logic over $L$ is the logic PPL, where the probabilistic operators may be iterated (nested). For example, in PPL we can express a statement like “this coin is counterfeit with probability 0.6”. In this paper we investigate the influence of probabilistic operators in the complexity of satisfiability in PL and PPL. We obtain complexity bounds, for the aforementioned satisfiability problem, which are parameterized in the complexity of satisfiability of conjunctions of positive and negative formulas that have neither a probabilistic nor a classical operator as a top-connective. As an application of our results we obtain tight complexity bounds for the satisfiability problem in PL and PPL when $L$ is classical propositional logic or justification logic.

Acknowledgements. The author is grateful to Antonis Achilleos and Thomas Studer for several useful discussions and suggestions and to the anonymous reviewers for many valuable comments and remarks that helped him improve the quality of this paper substantially. The initial part of this research was conducted when the author was at the Institute of Computer Science (INF) in Bern, Switzerland (financial support by the Swiss National Science Foundation project 153169, Structural Proof Theory and the Logic of Proofs) and the final part when the author was at the Lorraine Research Laboratory in Computer Science and its Applications (LORIA) in Nancy, France (financial support by the European Research Council project 313360, Epistemic Protocol Synthesis).
1 Introduction

1.1 Background and Related Work

Probabilistic logics (also known as probability logics) are logics that can be used to model uncertain reasoning. Although the idea of probabilistic logic was first proposed by Leibnitz, the modern development of this topic started only in the 1970s and 1980s in the papers of H. Jerome Keisler [14] and Nils Nilsson [24]. Following Nilsson’s research, Fagin, Halpern and Megiddo [9] introduced a logic with arithmetical operations built into the syntax so that Boolean combinations of linear inequalities of probabilities of formulas can be expressed. Based on Nilsson’s research Ognjanović, Rašković and Marković [26] defined the logic $LPP_2$, which is a non-iterated probabilistic logic with classical base. The language of $LPP_2$ is defined by adding (non-nested) operators of the form $P_{\geq s}$ (where $s$ is a rational number) in the language of classical propositional logic. In $LPP_2$ we can have expressions of the form $P_{\geq s} \alpha$, which read as “the probability of truthfulness of classical propositional formula $\alpha$ is at least $s$”. In addition to $LPP_2$, the authors of [26] define the logic $LPP_1$, which is a probabilistic logic over classical propositional logic, that allows iterations (nesting) of the probabilistic operators ($P_{\geq s}$). In $LPP_1$ we can describe a situation like the following: let $c$ be a coin and let $p$ be the event “$c$ lands tails”. Assume that the probability of $c$ landing tails is at least 60% (because $c$ is counterfeit). We can express this fact in non-iterated probabilistic logics by the formula $P_{\geq 0.6} p$. Assume now that we are uncertain about the fact of $c$ being counterfeit. In order to express this statement we need nested applications of the probabilistic operators. In $LPP_1$ for example we can have a formula like $P_{\geq 0.8} P_{\geq 0.6} p$.

In addition to classical propositional logic, probabilistic logics have been defined over several other logics (see the recent [27] for an overview). For example in [17, 18] we defined two probabilistic logics over justification logic (probabilistic justification logics for short). Justification logic [2] can be understood as an explicit analogue of modal logic [5]. Whereas traditional modal logic uses formulas of the form $\Box \alpha$ to express that an agent believes $\alpha$, the language of justification logic ‘unfolds’ the $\Box$-modality into a family of so-called justification terms, which are used to represent evidence for the agent’s belief. Hence, instead of $\Box \alpha$, justification logic includes formulas of the form $t : \alpha$, where $t$ is a justification term. Formulas of the form $t : \alpha$ are called justification assertions and read as:

the agent believes $\alpha$ for reason $t$.

Justification terms can represent any kind of objects that we use as evidence: for example proofs in Peano arithmetic or informal justifications (like everyday observations, texts in newspapers, or someone’s words). Artemov developed the first justification logic, the Logic of Proofs (usually abbreviated as LP), to provide intuitionistic logic with a classical provability semantics [3, 4]. Except from LP, several other justification logics have been introduced. The minimal justification logic is called $J$ [2]. By the famous realization theorem [4, 6] $J$
corresponds to the minimal modal logic $K$. That is, we can translate any theorem of $J$ to a theorem of $K$ by replacing any term with the $\Box$ and also any theorem of $K$ to a theorem of $J$ by replacing any occurrence of $\Box$ with an appropriate justification term.

The non-iterated probabilistic logic over $J$, the logic $PJ$, is defined in [17] and the iterated probabilistic logic over $J$, the logic $PPJ$, is defined in [18]. In $PJ$ we can describe a situation like the following: assume that an agent reads in some reliable newspaper that fact $\alpha$ holds and also that the agent hears that fact $\alpha$ holds from some unreliable neighbour. Then, the agent has two justifications for $\alpha$: the text of the newspaper, represented as $s$, and the words of their neighbour, represented as $t$. We can express the fact that the newspaper is a more reliable source than the neighbour using the $PJ$-formulas $P_{\geq 0.8}s : \alpha$ and $P_{\geq 0.2}t : \alpha$. So, we can use probabilistic justification logic to model the idea that:

different kinds of evidence for $\alpha$ lead to different degrees of belief in $\alpha$.

In $PPJ$ we can express our uncertainty for a justification that proves that a coin is counterfeit, e.g. using the formula $P_{\geq 0.8}P_{\geq 0.3}(t : p)$ where $p$ is a formula that states that a coin is counterfeit and $t$ is a justification for this fact. A more interesting property of $PPJ$ is the fact that the language of $PPJ$ allows applying justification terms to probabilistic operators and vice versa (as we will see later this is the property that makes finding complexity bounds for the satisfiability problem in $PPJ$ a challenging task). In [18] we have used this property to show that the famous lottery paradox [22] can be analysed in $PPJ$. The lottery paradox goes as follows: assume that we have 1,000 tickets in a lottery where every ticket has the same probability to win and also which has exactly one winning ticket. Now assume a proposition is believed if and only if its degree of belief is greater than 0.99. In this setting it is rational to believe that ticket 1 does not win, it is rational to believe that ticket 2 does not win, and so on. However, this entails that it is rational to believe that no ticket wins because rational belief is closed under conjunction. Hence, it is rational to believe that no ticket wins and that one ticket wins, which is absurd. In [18] we have formalized the lottery paradox in $PPJ$ and we have also proposed a solution for avoiding the paradox via restricting the axioms that are justified in $PPJ$.

A model for a non-iterated probabilistic logic is a probability space where the events are models of the base logic. A model for an iterated probabilistic logic is a probability space where the events contain models of the base logic and other probability spaces, so that we can deal with iterated probabilities. One can say that the models for iterated probabilistic logics look like Kripke structures, where the accessibility relation is replaced by a probability measure. The satisfiability problem for a probabilistic logic is to decide whether such a model that satisfies a given formula exists. In the 1980s Georgakopoulos et al. [11] studied a problem that is very similar to the satisfiability problem in probabilistic logics. This problem is called $PSAT$ and it is a probabilistic version of the famous satisfiability problem problem in classical propositional logic ($SAT$). The problem $PSAT$ can be formalized as follows: assume that we are given a
formula in conjunctive normal form and a probability for each clause. Is there a probability distribution (over the set of all possible truth assignments of the variables appearing in the clauses) that satisfies all the clauses? Georgakopoulos et al. reduced PSAT to solving a linear system, and proved that PSAT is NP-complete. Although the expressive power of the formal systems of [9] and [26] is richer than the one of [11], the authors of [9] and [26] were able to use similar arguments with the authors of [11] to show that the satisfiability problem in their logics is also NP-complete. In [15] we obtained tight bounds for the complexity of the satisfiability problem in non-iterated probabilistic justification logic, using again some results from the theory of linear programming. Fagin and Halpern [10] mention that complexity bounds for the satisfiability problem in a modal logic that allows nesting of the probabilistic operators (like in LPP₁) can be obtained by employing an algorithm based on a tableau construction as in classical modal logic [12]. In [16] we used the idea of Fagin and Halpern in order to obtain tight bounds for the complexity of satisfiability in PPJ.

1.2 Our Contribution

The goal of this paper is to summarize and generalize the results for the complexity of the satisfiability problem in non-iterated and iterated probabilistic logics. This paper is the extended journal version of [15] and of [16] which were presented in FoIKS 2016 and in the 11th Panhellenic Symposium in 2017 respectively. The results of [15, 16] refer only to probabilistic justification logic, whereas in the present paper we make clear that our results can be applied to a non-iterated and iterated probabilistic logic over any extension of classical propositional logic. We present upper and lower complexity bounds, for the aforementioned satisfiability problem, which are parameterized on the complexity of satisfiability of conjunctions of positive and negative formulas that have neither a probabilistic nor a classical operator as a top-connective. We also show how our results can be applied to the special cases where the probabilistic logics are defined over classical propositional logic or justification logic.

1.3 Outline of the Paper

In Section 2 we give some preliminary definitions and prove a lemma from the theory of linear programming that is necessary for our analysis. In Sections 3 and 4 we obtain complexity bounds for the satisfiability problem in non-iterated and iterated probabilistic logics over an extension of classical propositional logic. In Section 5 we apply the results of Sections 3 and 4 to determine the complexity of satisfiability in probabilistic logics over classical propositional logic and over justification logic. In Section 6 we make some final remarks and present an interesting open problem.
2 Preliminaries

For the purposes of this paper a logic is a formal system, defined via a set of axioms and inference rules, a notion of semantics (i.e. a formal definition of the notion of model for the logic) together with a provability and satisfiability relation over some formal language. In this paper we are interested in obtaining complexity bounds for the following decision problem:

**Definition 1** (Satisfiability Problem). Let \( L \) be a logic over some language \( \mathcal{L} \). The satisfiability problem for \( L \) (denoted as \( L_{\text{SAT}} \)) is the following decision problem:

> given some \( \alpha \in L \), is there a model of \( L \) that satisfies \( \alpha \)?

For a formula \( \alpha \) in the language of some logic \( L \), \( \alpha \) is satisfiable means that there is an \( L \)-model that satisfies \( \alpha \). If the satisfiability in \( L \) is defined in worlds of the models, then \( \alpha \) is satisfiable means that there is a model of \( L \), \( M \), and a world \( w \), such that \( \alpha \) is satisfied in the world \( w \) of \( M \). Since the satisfiability problem depends only on semantical notions, we will present all the logics without the corresponding axiomatization. The only exceptions are the basic justification logic \( J \) and the iterated probabilistic logic over \( J \), \( \text{PPJ} \), where it is necessary to know what the axioms of the logic are, for properly defining the models. We note that our results are independent of which the axioms of \( J \) and \( \text{PPJ} \) are (and also of whether the used notion of semantics is sound and complete with respect to these axioms), as long as these axioms are finitely many.

In the rest of the paper we fix a logic \( L \) over a language \( \mathcal{L} \). We assume that \( L \) is an extension of classical propositional logic and that \( \mathcal{L} \) is defined by the following grammar:

\[
\alpha ::= b(\alpha) \mid \neg \alpha \mid \alpha \land \alpha
\]

where \( b(\alpha) \in B(\mathcal{L}) \). \( B(\mathcal{L}) \) contains the formulas of \( \mathcal{L} \) that do not have \( \neg \) or \( \land \) as their top-connectives. We assume that \( B(\mathcal{L}) \) contains at least \( \text{Prop} \), which is a countable set of atomic propositions. We will refer to the elements of \( B(\mathcal{L}) \) as the **basic formulas** of language \( \mathcal{L} \) or the basic formulas of logic \( L \). We assume that we are given a function \( v \) which gives semantics to the elements of language \( \mathcal{L} \). In other words we assume that \( v \) assigns a truth value (T for true and F for false) to elements of \( B(\mathcal{L}) \). The extension of \( v \) to the elements of \( \mathcal{L} \) is the function \( \bar{v} \), which is defined classically. We will refer to \( v \) (or its extension) as an **evaluation**. We use Greek lower-case letters like \( \alpha, \beta, \gamma, \ldots \) for members of \( \mathcal{L} \). The symbol \( \mathcal{P} \) stands for powerset. We also define the following abbreviations in the standard way:

\[
\alpha \lor \beta \equiv \neg(\neg \alpha \land \neg \beta) ; \\
\alpha \rightarrow \beta \equiv \neg \alpha \lor \beta .
\]

From the above discussion it is clear that in order to define the semantics of \( L \) it suffices to determine which are the basic formulas and how the evaluation behaves on them. For example, if we assume that the basic formulas are atomic
propositions (i.e. elements of Prop) and that the evaluation is a truth assignment, then we have defined the language and semantics of classical propositional logic.

In the next sections we will define probabilistic logics over L. The models for these probabilistic logics will be probability spaces where the events are models for L (and in iterated case contain other probability spaces too). In order to formally present these models, we need the following definitions:

**Definition 2 (σ-Algebra Over a Set).** Let W be a non-empty set and let H be a non-empty subset of P(W). We call H a σ-algebra over W iff the following hold:

- W ∈ H;
- U ∈ H → W \ U ∈ H.
- For any countable collection of elements of H, U_0, U_1, ..., it holds that:
  \[ \bigcup_{i \in \mathbb{N}} U_i \in H. \]

**Definition 3 (σ-Additive Measure).** Let H be a σ-algebra over W and assume that \( \mu : H \to [0,1] \). We call \( \mu \) a σ-additive measure iff the following hold:

1. \( \mu(W) = 1. \)
2. Let \( U_0, U_1, ... \) be a countable collection of pairwise disjoint elements of H. Then:
   \[ \mu \left( \bigcup_{i \in \mathbb{N}} U_i \right) = \sum_{i \in \mathbb{N}} \mu(U_i). \]

**Definition 4 (Probability Space).** A probability space is a structure \( \langle W, H, \mu \rangle \), where:

- W is a non-empty set;
- H is a σ-algebra over W;
- \( \mu : H \to [0,1] \) is a σ-additive measure.

The members of H are called measurable sets.

A finitely additive measure can be defined by assuming a finite, instead of a countable union, in the previous definitions. Semantics for probabilistic logics over classical propositional logic has been given both for σ- and for finitely additive measures [26]. Semantics for probabilistic logics over justification logic [17, 18] has been given only for finitely additive measures. However, after the small model theorems that we will prove, the probability spaces in the models will be finite, so the results of this paper do not depend on the the measures being σ- or finitely additive.

As we mentioned in the introduction decidability and complexity results in probabilistic logics heavily depend on results from the theory of linear programming. In this paper we will use a theorem that provides bounds on the size of
a solution of a linear using the sizes of the constants that appear in the system. Before showing this result, we need to define the size for non-negative integers and rational numbers and to present Theorem 6. We use \textbf{bold} font for vectors. The superscript * in a vector denotes that the vector represents a solution of some linear system.

**Definition 5 (Sizes).** Let $r$ be a non-negative integer. The size of $r$, represented as $|r|$, is the number of bits needed for representing $r$ in the binary system. If $r = \frac{s_1}{s_2}$ is a rational number, where $s_1$ and $s_2$ are relatively prime non-negative integers with $s_2 \neq 0$, then the size of $r$ is $|r| := |s_1| + |s_2|$.

**Theorem 6** ([7, p. 145]). Let $S$ be a system of $r$ linear equalities. Assume that the vector $x^*$ is a solution of $S$ such that all of $x^*$’s entries are non-negative. Then there is a vector $y^*$ such that:

1. $y^*$ is a solution of $S$;
2. all the entries of $y^*$ are non-negative;
3. at most $r$ entries of $y^*$ are positive.

Theorem 7 provides the announced bounds on the solution of a linear system. A sketch of its proof was given in [9, Lemmata 2.5 and 2.7]. To make our presentation complete, we provide a detailed proof here.

**Theorem 7.** Let $S$ be a linear system of $n$ variables and of $r$ linear equalities and/or inequalities with integer coefficients each of size at most $l$. Assume that the vector $x^* = x_1^*, \ldots, x_n^*$ is a solution of $S$ such that for all $i \in \{1, \ldots, n\}$, $x_i^* \geq 0$. Then, there is a vector $y^* = y_1^*, \ldots, y_n^*$ that satisfies the following properties:

1. $y^*$ is a solution of $S$;
2. at most $r$ entries of $y^*$ are positive;
3. for all $i$, $y_i^*$ is a non-negative rational number with size bounded by

$$2 \cdot (r \cdot l + r \cdot \log_2(r) + 1).$$

**Proof.** We make the following conventions:

- All vectors used in this proof have $n$ entries. The entries of the vectors are assumed to be in one to one correspondence with the variables that appear in the original system $S$.
- Let $y^*$ be a solution of a linear system $T$. If $y^*$ has more entries than the variables of $T$ we imply that entries of $y^*$ that correspond to variables appearing in $T$ compose a solution of $T$.
- Assume that system $T$ has less variables than system $T'$. When we say that any solution of $T$ is a solution of $T'$ we imply that the missing variables are set to 0.
Assume that the original system $\mathcal{S}$ contains an inequality of the form:

$$b_1 \cdot x_1 + \ldots + b_n \cdot x_n \odot c,$$

for $\odot \in \{\prec, \leq, \geq, \succ\}$ where $x_1, \ldots, x_n$ are variables and $b_1, \ldots, b_n, c$ are constants that appear in $\mathcal{S}$. Vector $\mathbf{x}^*$ is a solution of (1). We replace the inequality (1) in $\mathcal{S}$ with the following equality:

$$b_1 \cdot x_1 + \ldots + b_n \cdot x_n = b_1 \cdot x_1^* + \ldots + b_n \cdot x_n^*.$$

We repeat this procedure for every inequality of $\mathcal{S}$. This way we obtain a system of linear equalities which we call $\mathcal{S}_0$. It is easy to see that $\mathbf{x}^*$ is a solution of $\mathcal{S}_0$ and that any solution of $\mathcal{S}_0$ is also a solution of $\mathcal{S}$.

Now we will transform $\mathcal{S}_0$ to another linear system by applying the following algorithm:

(i) Set $i := 0$, $e_0 := r$, $v_0 := n$, $\mathbf{x}^*_{\cdot 0} := \mathbf{x}^*$. Go to step (ii).

(ii) If $e_i = v_i$ then go to step (iii). Otherwise go to step (iv).

(iii) If the determinant of $\mathcal{S}_i$ is non-zero then stop. Otherwise go to step (vi).

(iv) If $e_i < v_i$ then go to step (v), else go to step (vi).

(v) We know that the vector $\mathbf{x}^{\cdot i}$ is a non-negative solution for the system $\mathcal{S}_i$.

From Theorem 6 we obtain a solution $\mathbf{x}^{\cdot i+1}$ for the system $\mathcal{S}_i$ which has at most $e_i$ entries positive. In $\mathcal{S}_i$ we replace the variables that correspond to zero entries of the solution $\mathbf{x}^{\cdot i+1}$ with zeros. We obtain a new system which we call $\mathcal{S}_{i+1}$ with $e_{i+1} = e_i$ equalities and $v_{i+1} = e_i < v_i$ variables. Vector $\mathbf{x}^{i+1}$ is a solution of $\mathcal{S}_{i+1}$ and any solution of $\mathcal{S}_{i+1}$ is a solution of $\mathcal{S}_i$. We set $i := i + 1$ and we go to step (ii).

(vi) We remove only one equation that can be written as a linear combination of some others. We obtain a new system which we call $\mathcal{S}_{i+1}$ with $e_{i+1} = e_i - 1$ equalities and $v_{i+1} = v_i$ variables. We set $i := i + 1$ and $\mathbf{x}^{\cdot i+1} := \mathbf{x}^{\cdot i}$. We go to step (ii).

From steps (v) and (vi) it is clear that during the execution of the above algorithm, the sum of the number of variables and equations decreases. Therefore, the algorithm terminates.

Let $I$ be the final value of $i$ after the execution of the algorithm. Since the only way for our algorithm to terminate is through step (iii) it holds that system $\mathcal{S}_I$ is an $e_I \times e_I$ system of linear equalities with non-zero determinant (for $e_I \leq r$). System $\mathcal{S}_I$ is obtained from system $\mathcal{S}_0$ by possibly replacing some variables that correspond to zero entries of the solution with zeros and by possibly removing some equalities (that have a linear dependence on others). So, any solution of $\mathcal{S}_I$ is also a solution of $\mathcal{S}_0$ and thus a solution of $\mathcal{S}$. From the algorithm we have that $\mathbf{x}^{\cdot I}$ is a solution of $\mathcal{S}_I$. Since $\mathcal{S}_I$ has a non-zero determinant Cramer’s rule
can be applied. Hence, the vector \( x^* \) is the unique solution of system \( S \). Let \( x^*_{i} \) be an entry of \( x^* \). Entry \( x^*_{i} \) is equal to the following rational number:

\[
\left| \begin{array}{ccc}
    a_{11} & \ldots & a_{1c_I} \\
    \vdots & & \vdots \\
    a_{e_I 1} & \ldots & a_{e_I c_I} \\
    b_{11} & \ldots & b_{1c_I} \\
    \vdots & & \vdots \\
    b_{e_I 1} & \ldots & b_{e_I c_I}
\end{array} \right|,
\]

where all the \( a_{ij} \) and \( b_{ij} \) are integers that appear in the original system \( S \). By properties of the determinant we know that the numerator and the denominator of the above rational number will each be at most equal to \( r! \cdot (2^l - 1)^r \). So we have that:

\[
| x^*_{i} | \leq 2 \cdot \left( \log_2 (r! \cdot (2^l - 1)^r) + 1 \right) \\
| x^*_{i} | \leq 2 \cdot \left( \log_2 (r^r \cdot 2^{2^l}) + 1 \right) \\
| x^*_{i} | \leq 2 \cdot \left( r \cdot \log_2 (r) + l \cdot r + 1 \right).
\]

As we already mentioned the final vector \( x^* \) is a solution of the original linear system \( S \). We also have that all the entries of \( x^* \) are non-negative, at most \( r \) of its entries are positive and the size of each entry of \( x^* \) is bounded by \( 2 \cdot (r \cdot \log_2 r + r \cdot l + 1) \). So, \( x^* \) is the desired vector \( y^* \).

\[\blacksquare\]

3 Non-Iterated Probabilistic Logics

In Subsection 3.1 we define the semantics for non-iterated probabilistic logics. In Subsection 3.2 we prove a small model property and in Subsection 3.3 we present the complexity bounds. The upper bound is obtained by guessing the small model and the lower bound via a reduction from the satisfiability problem in classical propositional logic.

3.1 Semantics

The non-iterated probabilistic logic over \( L \) is the logic \( PL \). The language of \( PL \), represented as \( L_{PL} \), is defined by adding non-nested probabilistic operators to the language \( L \). Formally, \( L_{PL} \) is described by the following grammar:

\[
A ::= P_{\geq s} \alpha \mid \neg A \mid A \land A
\]

where \( s \in \mathbb{Q} \cap [0,1] \) and \( \alpha \in L \). We define \( B(L_{PL}) := B(L) \), i.e. \( L_{PL} \) has the same basic formulas as \( L \). The intended meaning of the formula \( P_{\geq s} \alpha \) is that “the probability of truthfulness for \( \alpha \) is at least \( s \)”. For \( L_{PL} \), we assume the same abbreviations as for \( L \). The operator \( P_{\geq s} \) is assumed to have greater
precedence than all the connectives of \( \mathcal{L} \). We also define the following syntactical abbreviations:

\[
P_{\leq s} \alpha \equiv \neg P_{\geq s} \alpha ;
\]

\[
P_{\leq s} \alpha \equiv P_{\geq 1 - s} \neg \alpha ;
\]

\[
P_{> s} \alpha \equiv \neg P_{\leq s} \alpha ;
\]

\[
P_{=} \alpha \equiv P_{\geq s} \alpha \land P_{\leq s} \alpha .
\]

We use capital Latin letters like \( A, B, C, \ldots \) for members of \( \mathcal{L}_{PL} \) possibly primed or with subscripts.

A model for \( \mathcal{PL} \) is a probability space, where the events (also called worlds) are models for \( \mathcal{L} \). In order to determine the probability of truthfulness for an \( \mathcal{L} \)-formula \( \alpha \) in such a probability space we have to find the measure of the set containing all \( \mathcal{L} \)-models that satisfy \( \alpha \). More formally, we have:

**Definition 8 (PL-Model).** Let \( M = \langle W, H, \mu, v \rangle \) where:

- \( \langle W, H, \mu \rangle \) is a probability space;
- \( v \) is a function that assigns an evaluation to every \( w \) in \( W \). We write \( v_w \) instead of \( v(w) \).

\( M \) is a PL-model iff \( [\alpha]_M \in H \) for every \( \alpha \in \mathcal{L} \), where:

\[
[\alpha]_M = \{ w \in W \mid v_w(\alpha) = T \} .
\]

We will drop the subscript \( M \), i.e. we will simply write \([\alpha]\), if this causes no confusion.

**Definition 9 (Satisfiability in a PL-model).** Let \( M = \langle W, H, \mu, v \rangle \) be a PL-model. The satisfiability of \( \mathcal{L}_{PL} \)-formulas that have a probabilistic operator as their top-connective is defined as follows (the formulas with top-connectives \( \neg \) and \( \land \) are treated classically):

\[
M \models P_{\geq s} \alpha \iff \mu([\alpha]_M) \geq s .
\]

We observe that the semantics of \( \mathcal{PL} \) only depends on the definition of the evaluation.

### 3.2 Small Model Property

In this subsection we show that if \( A \in \mathcal{PL} \) is satisfiable then it is satisfiable in model that is polynomial in the size of \( A \). After we have this, it will be easy to obtain the upper complexity bound: we can simply guess the model in polynomial time and then, with the help of some oracles, verify that it satisfies \( A \). We have to draw the attention of the reader to the fact that the small model has to be small not only in terms of possible worlds, but also in terms of the probabilities that are assigned to each world. Otherwise we cannot show that we can guess the model in polynomial time.
At first we need some definitions. The set of subformulas of some formula \( A \), represented as \( \text{subf}(A) \), is defined as usual. The size of \( A \), represented as \(|A|\), is the number of symbols needed to write \( A \). In order to compute \(|A|\), the size of every probabilistic operator counts as one. So, \(|\neg P_{\geq \frac{1}{2}}| = 3\). For \( A \in \mathcal{L}_{PL} \) we define:

\[
||A|| := \max \left\{ |s| \mid P_{\geq \delta} \alpha \in \text{subf}(A) \right\}.
\]

**Definition 10** (Conjunctions of Positive and Negative Basic Formulas). Let \( A \in \mathcal{L} \cup \mathcal{L}_{PL} \). The set of conjunctions of positive and negative basic formulas of \( A \) is the following set:

\[
\text{cpnb}(A) = \left\{ a \mid a \text{ is of the form } \bigwedge_{B \in \text{subf}(A) \cap B(\mathcal{L})} \pm B \right\},
\]

where \( \pm B \) denotes either \( B \) or \( \neg B \). The acronym \( \text{cpnb} \) stands for conjunction of positive and negative basic formulas. If \( a \in \text{cpnb}(A) \) for some \( A \) and there is no danger of confusion we may say that \( a \) is \( \text{cpnb} \)-formula. We use the lower-case Latin letter \( a \) for \( \text{cpnb} \)-formulas, possibly with subscripts.

Let \( A \) be of the form \( \bigwedge_i B_i \) or of the form \( \bigvee_i B_i \). Then \( C \in A \) means that for some \( i \), \( B_i \equiv C \).

Theorem 11 proves the announced small model property. Its an adaptation of the small model Theorem 2.6 of [9]. In [15] the proof of the small property unnecessarily depends on the completeness theorem for \( PJ \). In Theorem 11 we remedy this mistake.

**Theorem 11** (Small Model Property for \( PL \)). Let \( A \in \mathcal{L}_{PL} \). If \( A \) is \( PL \)-satisfiable then it is satisfiable in a \( PL \)-model \( M = \langle W, H, \mu, v \rangle \) such that:

1. \(|W| \leq |A|\);
2. \( H = \mathcal{P}(W)\);
3. For every \( w \in W \), \( \mu(\{w\}) \) is a non-negative rational number with size at most

\[
2 \cdot (|A| \cdot ||A|| + |A| \cdot \log_2(|A|) + 1);
\]
4. For every \( a \in \text{cpnb}(A) \), there exists at most one \( w \in W \) such that \( \bar{v}_w(a) = T \).

**Proof.** Let \( A \) be satisfiable in some \( PL \)-model. We divide the proof in two parts:

- we show that the satisfiability of \( A \) implies that a linear system \( S \) is satisfiable;
- we use a solution of \( S \) to define a model for \( A \) that satisfies the properties (1)–(4).
Finding the Satisfiable Linear System. Let \( M \) be a PL-model. By some propositional reasoning we can show that:

\[
M \models A \iff M \models \bigvee_{i=1}^{K} \bigwedge_{j=1}^{l_i} P_{\ominus s_{ij}} \left( \alpha^{ij} \right).
\]

(2)

for some \( K \) and \( l_i \)'s, such that for each \( i \) and for each \( j, \ominus s_{ij} \in \{\geq, <\} \) and \( \alpha^{ij} \) is a disjunction of elements of \( \text{cpnb}(A) \). Since \( A \) is satisfiable, Eq. (2) implies that there exists a PL-model \( M' = \langle W', H', \mu', v' \rangle \) and some \( 1 \leq i \leq K \) such that:

\[
M' \models \bigwedge_{j=1}^{l_i} P_{\ominus s_{ij}} \left( \alpha^{ij} \right).
\]

(3)

Let \( \text{cpnb}(A) = \{a_1, \ldots, a_n\} \). For every \( k \in \{1, \ldots, n\} \) we define:

\[
x_k^* = \mu'([a_k]_{M'}) .
\]

(4)

In every world of \( M' \) some atom of \( A \) must hold. Thus, we have:

\[
\mu' \left( \bigcup_{k=1}^{n} [a_k]_{M'} \right) = 1 .
\]

(5)

The \( a_k \)'s are atoms of the same formula, so we have that for all \( k, k' \in \{1, \ldots, n\} \):

\[
k \neq k' \implies [a_k]_{M'} \cap [a_{k'}]_{M'} = \emptyset .
\]

(6)

By Eqs. (4),(5),(6) and the additivity of \( \mu' \) we get:

\[
\sum_{k=1}^{n} x_k^* = 1 .
\]

(7)

Let \( j \in \{1, \ldots, l_i\} \). From Eq. (3) we get \( M' \models P_{\ominus s_{ij}} \left( \alpha^{ij} \right) \). This implies that \( \mu'(\alpha^{ij}) \ominus s_{ij} \), i.e.

\[
\mu' \left( \bigvee_{a_k \in \alpha^{ij}} a_k \right)_{M'} \ominus s_{ij},
\]

from which we can show that:

\[
\mu' \left( \bigcup_{a_k \in \alpha^{ij}} [a_k]_{M'} \right) \ominus s_{ij} .
\]

By Eq. (4), (6) and the additivity of \( \mu' \) we have that:

\[
\sum_{a_k \in \alpha^{ij}} x_k^* \ominus s_{ij} .
\]
So we have that

$$\sum_{a_k \in \alpha_{ij}} x_k^* \odot_{ij} s_{ij}.$$ \hfill (8)

By Eqs. (4), (7) and (8) it is clear that the vector $x^* = x_1^*, \ldots, x_n^*$ is a non-negative solution of a linear system, call it $\mathcal{S}$. By Theorem 7 we have that there exists a vector $y^* = y_1^*, \ldots, y_n^*$, with non-negative entries, that is a solution of $\mathcal{S}$ and has at most $N$ entries positive, where $0 < N \leq |A|$. Without loss of generality we assume that $y_1^*, \ldots, y_N^*$ are the positive entries of $y^*$. Since every $x_k^*$ corresponds to a $\text{cpnb}$-formula of $A$ we can associate every positive $y_k^*$ with the satisfiable atom $a_k$.

**Defining the Model that Satisfies $A$.** The quadruple $M = \langle W, H, \mu, v \rangle$ is defined as follows:

(a) $W = \{w_1, \ldots, w_N\}$, for some $w_1, \ldots, w_N$;

(b) $H = \mathcal{P}(W)$;

(c) For all $V \in H$:

$$\mu(V) = \sum_{w_k \in V} y_k^*;$$

(d) Let $i \in \{1, \ldots, N\}$. $v_{w_i}$ is an evaluation that satisfies $a_i$.

By using the fact that each $y^*$ is a solution of $\mathcal{S}$ we can show that $M$ is a $\mathcal{PL}$-model. We will now prove the following statement:

$$\forall 1 \leq k \leq N \left[ w_k \in [\alpha_{ij}]_M \iff a_k \in \alpha_{ij} \right].$$ \hfill (9)

Let $k \in \{1, \ldots, n\}$. We prove the two directions of Eq. (9) separately.

($\Rightarrow$) Assume that $w_k \in [\alpha_{ij}]$. This means that $v_{w_k}(\alpha_{ij}) = T$. Assume that $a_k \notin \alpha_{ij}$. Then, since $\alpha_{ij}$ is a disjunction of some atoms of $A$, there must exist some $a_{k'} \in \alpha_{ij}$, with $k \neq k'$, such that $v_{w_k}(a_{k'}) = T$. However, by definition we have that $v_{w_k}(a_k) = T$. But this is a contradiction, since $a_k$ and $a_{k'}$ are different atoms of the same formula, which means that they cannot be satisfied in the same evaluation. Hence, $a_k \in \alpha_{ij}$.

($\Leftarrow$) Assume that $a_k \in \alpha_{ij}$. We know that $v_{w_k}(a_k) = T$, which implies that:

$$v_{w_k}(\alpha_{ij}) = T,$$

i.e. $w_k \in [\alpha_{ij}]_M$.

Hence, Eq. (9) holds. Now, we will prove the following statement:

$$\forall 1 \leq j \leq l_i \left[ M \models P_{\odot_{ij} s_{ij}} \alpha_{ij} \right].$$ \hfill (10)
Let $j \in \{1, \ldots, l_i\}$. It holds
\[
M \models P_{\otimes_j s_{ij}}(\alpha^{ij}) \iff \mu([\alpha^{ij}]_M) \otimes_j s_{ij} = \sum_{w_k \in [\alpha^{ij}]_M} y_k^* \otimes_j s_{ij} \iff \sum_{a_k \in \alpha^{ij}} y_k^* \otimes_j s_{ij}.
\]
The last statement holds because $y^*$ is a solution of $S$. Thus, Eq. (10) holds.

By Eq. (10) we have that $M \models \bigwedge_{j=1}^{l_i} P_{\otimes_j s_{ij}}(\alpha^{ij})$, which implies that
\[
M \models \bigvee_{i=1}^K \bigwedge_{j=1}^{l_i} P_{\otimes_j s_{ij}}(\alpha^{ij}),
\]
which, by Eq. (2), implies that $M \models A$.

So, we have that each $w_i$ corresponds to one satisfiable $a_i$ and also that $\mu(\{w_i\}) = y^*_i$. Since the number of positive $y^*_i$’s is at most $|A|$ and the size of every positive $y^*_i$ is at most $2 \cdot (|A| \cdot ||A|| + |A| \log_2(|A|) + 1)$, we have that $M$ is the model in question.

\section{3.3 Complexity Bounds}

In this subsection we obtain upper and lower complexity bounds for $\text{PL}_{\text{SAT}}$. The upper bounds follow by the fact that for a given $\text{PL}$ formula $A$, we can guess a small model for it and then verify that this model indeed satisfies $A$. The lower bound follows by the fact that $L$ is an extension of classical propositional logic.

As a first step we need the following two Lemmata which can be proved by an easy induction on the complexity of the formula.

\begin{lemma}
Let $\alpha \in \mathcal{L}$ and let $a \in \text{cpnb}(\alpha)$. Let $v_1, v_2$ be two evaluations and assume that for every basic formula $\beta$ that appears in $\alpha$:
\[
v_1(\beta) = v_2(\beta).
\]
Then we have:
\[
\tilde{v}_1(\alpha) = \tilde{v}_2(\alpha).
\]
\end{lemma}

\begin{lemma}
Let $\alpha \in \mathcal{L}$ and let $a \in \text{cpnb}(\alpha)$. Let $v$ be an evaluation and assume that $\tilde{v}(a) = T$. The decision problem:
\[
does \tilde{v} satisfy \alpha?\]
belongs to the complexity class $P$.
\end{lemma}

Now we are ready to prove the upper complexity bound for $\text{PL}_{\text{SAT}}$.  

Theorem 14. Assume that the satisfiability problem for cpnb-formulas in the logic $L$ belongs to the complexity class $C$. Then $PL_{SAT} \in NP^C$.

Proof. Let $A \in L_{PL}$ and let $A$ be the $C$-algorithm that can test cpnb-formulas in $L$ for satisfiability. By the small model theorem for $PL$ it suffices to guess a model for $A$ that satisfies the conditions (1)–(4) that appear in the statement of Theorem 11. We present a non-deterministic algorithm that performs this guess and we evaluate its complexity.

Algorithm: We guess $n$ elements of $cpnb(A)$, call them $a_1, \ldots, a_n$, and we also choose $n$ worlds, $w_1, \ldots, w_n$, for $n \leq |A|$. Using $A$ we can verify that for each $i \in \{1, \ldots, n\}$ there exists an evaluation $v_i(a_i) = T$. We define $W = \{w_1, \ldots, w_n\}$ and for every $i \in \{1, \ldots, n\}$ we set $v_{w_i} = v_i$. Since we are only interested in the satisfiability of basic formulas that appear in $A$, by Lemma 12, the choice of the $v_{w_i}$ is not important (as long as $v_{w_i}$ satisfies $a_i$).

We assign to every $\mu(\{w_i\})$ a rational number with size at most:

$$2 \cdot (|A| \cdot ||A|| + |A| \cdot \log_2(|A|) + 1).$$

We set $H = \mathcal{P}(W)$ and for every $V \in H$ we set:

$$\mu(V) = \sum_{w_i \in V} \mu(\{w_i\}).$$

It is then straightforward to see that conditions (1)–(4) that appear in the statement of Theorem 11 hold.

Now we have to verify that our guess is correct, i.e. that $M \models A$. Assume that $P_{\geq s} \alpha$ appears in $A$. In order to see whether $P_{\geq s} \alpha$ holds we need to calculate the measure of the set $[\alpha]_M$ in the model $M$. The set $[\alpha]_M$ will contain every $w_i \in W$ such that $v_{w_i}(\alpha) = T$. Since $v_{w_i}$ satisfies an atom of $A$ it also satisfies an atom of $\alpha$. So, by Lemma 13, we can check whether $w_i$ satisfies $\alpha$ in polynomial time. If $\sum_{w_i \in [\alpha]_M} \mu(\{w_i\}) \geq s$ then we replace $P_{\geq s} \alpha$ in $A$ with the truth value $T$, otherwise with the truth value $F$. We repeat the above procedure for every formula of the form $P_{\geq s} \alpha$ that appears in $A$. At the end we have a formula that is constructed only from the connectives $\neg$, $\land$ and the truth constants $T$ and $F$. Obviously, we can verify in polynomial time that the formula is true. This, of course, implies that $M \models A$.

Complexity Evaluation:

All the objects that are guessed in our algorithm have size that is polynomial in the size of $A$. Also the verification phase of our algorithm can be made in polynomial time. Furthermore checking whether an element of $cpnb(\alpha)$ is satisfiable is possible with a $C$-oracle. Thus, our algorithm belongs to the class $NP^C$. ■

And now the lower bound.

Theorem 15. The decision problem $PL_{SAT}$ is NP-hard.
Proof. We draw a reduction from the satisfiability problem in classical propositional logic. Let $\alpha$ be a formula of classical propositional logic. Clearly we have that $P_{\geq 1}\alpha \in \mathcal{L}_{PL}$. We will prove that:

$\alpha$ is satisfiable $\iff P_{\geq 1}(\alpha)$ is $PL$-satisfiable.

$\implies$: Assume that there exists a truth assignment $v$, such that $\bar{v}(\alpha) = T$. Then we can construct the quadruple $M = \langle W, H, \mu, v' \rangle$ with

$W = \{w\}$; $H = \emptyset, \{w\}$; $\mu(\emptyset) = 0$; $\mu(\{w\}) = 1$;

$v'_w$ is an extension of $v$ to basic formulas of $\mathcal{L}$.

It is then straightforward to show that $M$ is a $PL$-model and that $M \models P_{\geq 1}\alpha$.

$\impliedby$: Assume that there exists a $PL$-model $M = \langle W, H, \mu, v \rangle$ such that $M \models P_{\geq 1}\alpha$, i.e. $\mu([\alpha]_M) \geq 1$.

If $[\alpha]_M = \emptyset$ then it should be $\mu([\alpha]_M) = 0$ which contradicts the fact that $\mu([\alpha]_M) \geq 1$. Hence, there is an evaluation that satisfies $\alpha$ and since $\alpha$ is a classical formula, this implies that there is a truth assignment that satisfies $\alpha$. $\blacksquare$

4 Iterated Probabilistic Logics

4.1 Semantics

The iterated probabilistic logic over $L$ is the logic $PPL$ (the two $P$’s stand for the iterations of the probability operator). The language of $PPL$, $\mathcal{L}_{PPL}$, is defined by adding nested probabilistic operators to the language $\mathcal{L}$. Formally, $\mathcal{L}_{PPL}$ is defined by the following grammar:

$A ::= b(A) \mid \neg A \mid A \land A \mid P_s A$,

where $s \in \mathbb{Q} \cap [0,1]$ and $b(A) \in \mathcal{B}(\mathcal{L}_{PPL})$. We will use upper-case latin letters like $A, B, C, \ldots$ for members of $\mathcal{L}_{PPL}$.

Models for $PPL$ are probability spaces where the worlds contain evaluations and probability spaces (so that we can deal with iterated probabilities). Formally, we have:

Definition 16 ($PPL$-Model). Let $M = \langle U, W, H, \mu, v \rangle$ where:

1. $U$ is a non-empty set of objects called worlds;

2. for every $w \in U$:

$W_w, H_w, \mu_w$ is a probability space with $W_w \subseteq U$ and $v_w$ is an evaluation.
$M$ is a PPL-model iff for every $A \in \mathcal{L}_{PPL}$ and every $w \in U$, $[A]_{M,w} \in H_w$, where

$$[A]_{M,w} = \{ u \in W_w \mid M, u \models A \}.$$

**Definition 17** (Satisfiability in a PPL-model). Let $M = \langle U, W, H, \mu, v \rangle$ be a PPL-model. Satisfiability is defined as follows (the connectives $\neg$ and $\land$ are treated classically):

$$M, w \models A \iff v_w(A) = T \text{ for } A \in \mathcal{B}(\mathcal{L}_{PPL}) ;$$

$$M, w \models P_{\geq s}A \iff \mu_w([A]_{M,w}) \geq s.$$

We observe that, as in the non-iterated case, the definition of a PPL-model depends only on the definition of the evaluation.

### 4.2 Complexity Bounds

In this section we obtain complexity bounds for $\text{PPL}_{\text{SAT}}$. The upper bound is obtained via a tableaux procedure, which resembles the corresponding tableaux procedure for modal logic [12]. The idea for obtaining this upper bound for a modal logic that contained probabilistic operators similar to ours was sketched in [10, Theorem 4.5]. The lower bound is obtained by drawing a reduction from modal logic $\text{D}$ [12].

#### 4.2.1 The Upper Bound

We will need the following definition:

**Definition 18** (Conjunctions of Positive and Negative Formulas). Assume that $A_1, \ldots, A_n \in \mathcal{L}_{PPL}$. We define the following set:

$$\text{cpnf}(A_1, \ldots, A_n) = \left\{ a \mid a \text{ is of the form } \bigwedge_{i=1}^n \pm A_i \right\}.$$  

The acronym $\text{cpnf}$ stands for conjunction of positive and negative formulas. As for the $\text{cpnb}$-formulas we will use the possibly primed or subscripted lower-case Latin letter $a$ for $\text{cpnf}$-formulas. If $a \in \text{cpnf}(A_1, \ldots, A_n)$ for some $A_1, \ldots, A_n$ and there is no danger of confusion, we may say that $a$ is $\text{cpnf}$-formula.

We now present the announced tableaux method. Our tableaux are trees where the nodes are formulas prefixed with world and truth signs. So, the node $w \top A$ ($w \bot A$) intuitively means that formula $A$ is true (respectively false) at world $w$ (of some model). The root of a tableau is the formula that is tested for satisfiability. The tableaux rules are presented in Table 1. The first line consists of the propositional rules and the second line of the probabilistic rule $\text{prob}$. A separator $|$ in the result of the rule means that the formulas in the conclusion belong to distinct branches. So, only the rules $\text{prob}$ and $\text{andF}$ create new branches; the other rules simply add formulas to the branch where the premise belongs.
Every propositional rule gives simpler conditions for satisfiability: if the premise is satisfiable then at least one of the results has to be satisfiable too. The function of rule \text{prob} is more complicated and requires some explanation. Rule \text{prob} is the only rule that creates new worlds. So, the formulas that belong in a path between two applications of rule \text{prob} are marked with the same world. Therefore, we can define the notion of a \textit{world path}. A world path is a path in a tableau that starts either from the root or from a result of an application of the rule \text{prob} and ends either in a leaf or at a premise of an application of the rule \text{prob}. Due to the fact that all the nodes in a world path are marked with the same world, we can represent a world path as \( w.1 \, T \, A \| \cdots \| \, w.n \, T \, a_n \). Assume that:

\[
\{B_1, \ldots, B_m\} = \{B \mid P_{\geq s} B \in p_w\}.
\]

Then the formulas \( a_i \) appearing in the result of rule \text{prob} are defined as follows:

\[
\{a_1, \ldots, a_n\} = \text{cpnf}(B_1, \ldots, B_m).
\]

A world path is called \( P \)-open if there is some \( P_{\geq s} B \in p_w \). Otherwise it is called \( P \)-closed. Let \( p_w \) be a \( P \)-open world path that ends in an application of rule \text{prob} which we call \( \rho \). All the world paths that start from a result of \( \rho \) are called the \textit{children} of \( p_w \). For making the presentation simpler, we will use the nodes as expressions in the metalanguage. So, when we simply write \( w.1 \, T \, A \), we imply that the node \( w.1 \, T \, A \) belongs to the tableau. For every \( p_w \) we define the following PPL-formulas:

\[
\begin{align*}
F_{p_w} & = \bigwedge_{w \, T \, B} B \land \bigwedge_{w \, F \, B} \neg B; \\
\text{P}_{p_w} & = \bigwedge_{w \, T \, B, \, B \in \mathcal{B}(L_{ppl})} B \land \bigwedge_{w \, F \, B, \, B \in \mathcal{B}(L_{ppl})} \neg B.
\end{align*}
\]

Observe that \( \text{P}_{p_w} \) is always a \textit{cpnb}-formula. Also if \( F_{p_w} \) is PPL-satisfiable this implies that there is an evaluation that satisfies \( P_{p_w} \).

The tableau for some \( A \in L_{ppl} \) is a tree that is created as follows:

1. Create the node \( w \, T \, A \) (this is the root of the tableau). Go to step 2.
2. Apply the propositional rules for as long as possible. If there exists a
$P$-open world path, go to step 3. Otherwise stop.

3. Apply the rule prob to every $P$-open world path. Go to step 2.

Every time a tableaux rule is applied, at least one operator ($\neg$, $\land$, or $P_e$) is eliminated. This implies that the tableau for $A$ is a finite tree.

Now we are ready to prove the main theorem of this section.

**Theorem 19.** Assume that the satisfiability problem for $\text{cpnb}$-formulas in the logic $\text{PPL}$ belongs to the complexity class $C$. Then $\text{PPLSAT} \in \text{PSPACE}^C$.

**Proof.** Let $A$ be the $\mathcal{L}_{\text{PPL}}$-formula that we want to test for satisfiability. Let $A$ be the $C$-algorithm that decides the satisfiability problem for $\text{cpnb}$-formulas in logic $\text{PPL}$. We present an algorithm that decides whether $A$ is satisfiable by traversing the tableau for $A$. Then, we prove the correctness of the algorithm and analyse its complexity.

**Algorithm.** The goal of the algorithm is to traverse the tableau for $A$ and decide which world paths will be marked realizable. A realizable world path $p_w$ contains all the formulas that are satisfied in world $w$ of the model for $A$ (if our algorithm decides that such a model exists). On the other hand, a world path that is not marked realizable implies that the formulas in this path cannot be satisfied in a $\text{PPL}$-model. We execute the following steps:

1. Let $p_w$ be the next (in depth first fashion) world path. Go to step 2.

2. If $p_w$ is $P$-closed and $A$ fails in $P_{p_w}$ then stop. If $p_w$ is $P$-closed and $A$ succeeds in $P_{p_w}$, then mark $p_w$ realizable. If $p_w$ is $P$-open then go to step 3.

3. Select at most $|A|$ positive rational numbers of size at most $2 \cdot (|A| \cdot ||A|| + |A| \cdot \log_2(|A|) + 1)$, which have size at most 1 and assign each one of them to a child of $p_w$. Assign the number 0 to the rest of $p_w$’s children. For each node of the form $w \land P_{\geq s} B$ find all the children of $p_w$ that contain $B$ prefixed with $\land$ and add the rational numbers assigned to them. If the number is less than $s$, then move to 1 (without marking $p_w$ realizable). If the sum is at least $s$ move to the next node of the form $w \land P_{\geq s} B$.

Similarly, check every node of the form $w \lor P_{\geq s} B$. Let $X$ be the set of all children of $p_w$ to which a positive rational number is assigned. If there exists one member of $X$ where step 2 or 3 of the algorithm fails then mark $p_w$ as unrealizable. If steps 2 and 3 succeed for every member of $X$ and $A$ succeeds in $P_{p_w}$, mark $p_w$ realizable and go to step 1.

If, at the end of the algorithm there exists a world path starting from the root, that is marked realizable, return “satisfiable”. Otherwise, return “not satisfiable”.

19
Correctness. In order to prove our algorithm correct it suffices to show that for every world path $p_w$:

$$p_w \text{ is marked realizable } \iff F_{p_w} \text{ is PPL-satisfiable}. \quad (11)$$

Let $p_w$ be a world path. We prove the two directions of (11) separately:

$\implies$: We define the structure $M = \langle U, W, H, \mu, v \rangle$ as follows:

- $U = \{ u \mid p_u \text{ is marked realizable in the subtree of the tableau that has the first node of } p_w \text{ as a root} \}$.

And for every $u \in U$:

- $W_u = U$ and $H_u = \mathcal{P}(W_u)$.
- For every $v$, such that $p_v$ is a child of $p_u$ we define $\mu_u(\{v\})$ to be the rational number assigned to $p_v$ (which can be 0). For every $v \in U$ such that $v$ is not a child of $p_u$, we set $\mu_u(\{v\}) = 0$.
- $v_u$ is the evaluation that satisfies $P_{p_u}$. We know that such an evaluation exists since $p_u$ can be marked satisfiable only if $A$ succeeds in $P_{p_u}$.

Since for every $u \in U$ the rational numbers assigned to $p_u$’s children were selected in such a way such that there sum equals 1, it is straightforward to show that $M$ is a PPL-model. We will now show that for every $B \in \text{subf}(A)$ and for every $u \in U$:

$$u \models B \implies M, u \models B \quad \text{and} \quad u \not\models B \implies M, u \not\models B. \quad (12)$$

We proceed by induction on $B$. The only interesting case is when $B \equiv P_{\geq s} C$. Assume that $u \models P_{\geq s} C$. Then we have:

$$\sum_{ \{p_v \mid p_v \text{ is a child of } p_u \text{ and } v \models C \} } r_v \geq s,$$

where $r_v$ is the rational number that is assigned to $p_v$. By i.h. and by the definition of $M$ we have:

$$\sum_{ \{v \mid v \in W_u \text{ and } M, v \models C \} } \mu_u(\{v\}) \geq s$$

which by the additivity of $\mu_u$ gives us:

$$\mu_u([C] M, u) \geq s$$

i.e.

$$M, u \models P_{\geq s} C.$$
Exactly the same arguments prove the right conjunct of (12) and this concludes
the proof for the direction \( \implies \) of (11).

\[ \iff: \] We prove the claim by induction on the depth of \( p_w \) in the tableau for
\( A \).

If \( p_w \) is a leaf, then \( p_w \) is \( P \)-closed, which implies that \( F_{p_w} \equiv P_{p_w} \). The fact
that \( F_{p_w} \) is PPL-satisfiable implies that there is an evaluation that satisfies \( P_{p_w} \).
This implies that \( A \) is successful on \( P_{p_w} \), so \( p_w \) is marked realizable.

Assume that \( p_w \) has children. Assume that \( F_{p_w} \) is satisfiable in world \( w \) of
the PPL-model \( M \). Since \( p_w \) is \( P \)-open there is a \( K \) and some \( C_i \)'s and \( s_i \)'s such
that \( M, w \models \bigwedge_i^K P_{\circ \iota_i} C_i \) for \( \circ \iota_i \in \{ \geq, < \} \). Let:

\[
\text{cpnf}(C_1, \ldots, C_K) = \{ a_1, \ldots, a_m \} .
\]

By propositional reasoning we can show that \( M, w \models \bigwedge_i^K P_{\circ \iota_i} D_i \) where every
\( D_i \) is equivalent to a disjunction of some \( a_j \)'s. Now we proceed as in the proof
of Theorem 11. We show that the fact that the \( P_{\circ \iota_i} D_i \)'s are satisfied in \( w \)
implies that there is a linear system \( S \) which has as a solution a vector, every
entry of which corresponds to the measure of some \( a_j \)'s. By Theorem 7 we can
show that at most \(|A| \) entries of a solution for \( S \) have to be positive. And each
of the entries has size at most \( 2 \cdot (|A| \cdot |A|) + |A| \cdot \log_2(|A|) + 1 \). Recall that
each \( a_i \) is assigned to a child of \( p_w \). So, there are at most \(|A| \) positive rational
numbers that are assigned to children of \( p_w \). The algorithm that traverses the
tableau for \( A \) should be able to find them. Then the algorithm should be able to
verify that these rational numbers sum to 1 and satisfy the nodes that contain
\( P_{\circ \iota_i} C_i \) in \( p_w \). Also the fact that \( F_{p_w} \) is satisfiable implies that \( A \) succeeds in
\( P_{p_w} \). Furthermore the \( a_j \)'s that correspond to positive measures are satisfiable
in \( M \). This implies that for every \( p_{w,i} \), if a positive rational number is assigned
to \( p_{w,i} \), then \( F_{p_{w,i}} \) is satisfied in \( M \). By i.h. we have that the children of \( p_w \)
that correspond to positive measures are marked realizable. We conclude that
\( p_w \) is marked realizable.

This concludes the proof for (11).

**Complexity Analysis.** We will show that our algorithm can decide whether
there exists a world path starting from \( A \) that should be marked realizable by
using only a polynomial number of bits and a \( C \)-oracle. We observe that the fact
whether \( p_w \) should be marked realizable only depends on \( p_w \) and the subtree
below it. So, we can traverse the tableau tree in depth first fashion, reusing
space. For every \( p_w \) we need a polynomial number of bits to store \( p_w \) itself, the
positive rational numbers assigned to some of its children and the \( \text{cpnf} \)-formulas
that are assigned to these children. We can verify (using the \( C \)-oracle) that \( P_{p_w} \)
is satisfiable and that the probabilistic constraints in \( p_w \) are satisfied. Then we
can move to the first child of \( p_w \) (among those to which a positive probability is
assigned) and repeat the same procedure. Clearly, once we have that the first
child of \( p_w \) is marked realizable we do not need the space used for this child
any more. So, this space can be used for the next child. We conclude that the
maximum number of information that we have to store each time is at most
equal to the depth of the tree (which is polynomial on \(|A|\)) times the number of
bits needed to process a single world path (which as we observed is polynomial on \(|A|\) again). We conclude that our algorithm runs in polynomial space using a \(\mathcal{C}\)-oracle.

4.2.2 The Lower Bound

Before showing the reduction from modal logic \(\mathcal{D}\) we observe that the tableau decision procedure implies a small model property for \(\mathsf{PPL}\).

**Corollary 20. (Small Model Property for \(\mathsf{PPL}\))** Let \(A \in \mathcal{L}_{\mathsf{PPL}}\) be satisfiable. Then \(A\) is satisfiable in a \(\mathsf{PPL}\)-model \(M = \langle U, W, H, \mu, v \rangle\), where:

1. \(U \leq 2^{|A|}\);
2. for each \(w \in U\):
   (a) \(W_w = U\) and \(H_w = \mathcal{P}(U_w)\);
   (b) for every \(V \in H_w: \mu_w(V) = \sum_{v \in V} \mu(\{v\})\).

**Proof.** The fact that \(A\) is satisfiable implies that the tableau procedure for \(A\) succeeds. So, if we start a tableau procedure with \(w \top A\) in the root we should find a realizable world path that starts from \(w \top A\). Then as in the proof of \(\implies\) of (11) we can construct a \(\mathsf{PPL}\)-model for \(A\) that satisfies the properties in the statement of this corollary. Since the tableau for \(A\) is finite, the size of the model satisfying \(A\) is finite. 

Now we proceed with some standard definitions from modal logic. The language of modal logic, \(\mathcal{L}_{\Box}\), is described by the following grammar:

\[
A ::= p \mid \neg A \mid A \land A \mid \Box A .
\]

where \(p \in \mathsf{Prop}\). A Kripke model is structure \(M = \langle W, R, v \rangle\) where \(W\) is a non-empty set of worlds, \(R \subseteq W \times W\) and \(v\) is a function that assigns a truth assignment (for classical propositional logic) to every world in \(W\). For each \(w \in W\) we define the following set:

\[
R[w] = \{ u \mid (w, u) \in R \} .
\]

A Kripke model \(M = \langle W, R, v \rangle\) is serial if for every \(w \in W\), \(R[w] \neq \emptyset\). \(\mathcal{D}\) is the modal logic that is sound and complete with respect to serial Kripke models [12].

**Theorem 21.** \(\mathsf{PPL}_{\mathsf{SAT}}\) is \(\mathsf{PSPACE}\)-hard.

**Proof.** We will reduce \(\mathsf{D}_{\mathsf{SAT}}\) to \(\mathsf{PPL}_{\mathsf{SAT}}\). Since \(\mathsf{D}_{\mathsf{SAT}}\) is \(\mathsf{PSPACE}\)-complete [12] our theorem follows. Let \(A \in \mathcal{L}_{\Box}\) and let \(f(A)\) be the \(\mathsf{PPL}\)-formula obtained from \(A\) by replacing every occurrence of \(\Box\) by \(P_{\geq 1}\). We will show the following equivalence:

\[
A \text{ is \(\mathcal{D}\)-satisfiable iff } f(A) \text{ is \(\mathsf{PPL}\)-satisfiable} .
\]
⇒: Assume that \( A \) is satisfiable. By the small model theorem for modal logic \( D \) [12] it is satisfiable in a serial Kripke model \( M_D = \langle U, R, v \rangle \), where \( U \) is finite. We define \( M_{PPL} = \langle U, W, H, \mu, v' \rangle \) where for every \( w \in U \):

\[
W_w = R[w] \\
H_w = P(W_w) \\
v'_w \text{ is an extension of } v_w \text{ to } B(L_{PPL})\)-formulas
\]

and for every \( V \in H_w \):

\[
\mu_w(V) = \frac{|V|}{|R[w]|}.
\]

It is easy to show that \( M_{PPL} \) is a PPL-model. We will now show that:

\[
(\forall w \in U)(\forall B \in \text{subf}(A))[M_D, w \models B \iff M_{PPL}, w \models f(B)]
\]

by induction on the complexity of \( B \). The only interesting case is when \( B \) is of the form \( \square C \). Then we have:

\[
M_D, w \models \square C \iff (\forall u \in R[w])\left[ M_D, u \models C \right] \quad \overset{i.h.}{\iff} \quad (\forall u \in W_w)\left[ M_{PPL}, u \models f(C) \right]
\]

Now we have that \( W_w = [f(C)]_{M_{PPL},w} \) which immediately implies that

\[
\mu_w([f(C)]_{M_{PPL},w}) = 1.
\]

On the other hand assume that \( \mu_w([f(C)]_{M_{PPL},w}) = 1 \). Then if \( [f(C)]_{M_{PPL},w} \not\subseteq W_w \) then, by the definition of \( \mu_w \) we get that \( \mu_w(W_w \setminus [f(C)]_{M_{PPL},w}) > 0 \), which, by the additivity of \( \mu_w \) contradicts the fact that \( \mu_w([f(C)]_{M_{PPL},w}) = 1 \). We conclude that (14) is equivalent to the following:

\[
\mu_w([f(C)]_{M_{PPL},w}) = 1 \iff \quad M_{PPL}, w \models P_{\geq 1}f(C) \iff \quad M_{PPL}, w \models f(B)
\]

⇐: Assume that \( f(A) \) is satisfiable. Then \( f(A) \) is satisfiable in a model \( M_{PPL} = \langle U, W, H, \mu, v' \rangle \) that satisfies the properties of Corollary 20. Let \( M_D = \langle U, R, v' \rangle \), where for every \( w \in W \),

\[
R[w] = \{ u \in W_w \mid \mu_w(\{u\}) > 0 \}
\]

and \( v'_w \) is a restriction of \( v_w \) to Prop. It is straightforward to show that \( M_D \) is a serial Kripke structure. We will now show that:

\[
(\forall w \in W)(\forall B \in \text{subf}(A))[M_{PPL}, w \models f(B) \iff M_D, w \models B]
\]

23
by induction on the complexity of $B$. The only interesting case is when $B \equiv \Box C$.

We have that

$$M_D, w \models \Box C \quad \iff \quad (\forall u \in R[w]) [M_D, u \models C] \quad \iff \quad (\forall u \in W_w) [\mu_w(\{u\}) > 0 \implies M_{\text{PPL}}, u \models f(C)]$$

(15)

So $[f(C)]_{MPPL, w} \supseteq \{ u \in W_w \mid \mu_w(\{u\}) > 0 \}$. Hence

$$\mu_w([f(C)]_{MPPL, w}) = \sum_{u \in [f(C)]_{MPPL, w}} \mu(\{u\}) = \sum_{u \in W_w \mid \mu_w(\{u\}) > 0} \mu(\{u\}) = 1.$$ 

On the other hand assume that $\mu_w([f(C)]_{MPPL, w}) = 1$. Let $u \in W_w$ such that $\mu_w(\{u\}) > 0$. Assume that $M_{\text{PPL}}, u \not\models f(C)$. Then $u \in W_w \setminus [f(C)]_{MPPL, w}$. So $\mu_w(W_w \setminus [f(C)]_{MPPL, w}) > 0$, which contradicts the fact that $\mu_w([f(C)]_{MPPL, w}) = 1$. So for all $u \in W_w$, $\mu_w(\{u\}) > 0$ implies that $M_{\text{PPL}}, u \models f(C)$. We conclude that (15) is equivalent to:

$$\mu_w([f(C)]_{MPPL, w}) = 1 \quad \iff \quad M_{\text{PPL}}, w \models P_{\geq 1} f(C) \quad \iff \quad M_{\text{PPL}}, w \models f(B).$$

We conclude that (13) holds, which proves our theorem. \[\blacksquare\]

5 Applications

In this section we apply the results of Sections 3 and 4 in probabilistic logics over classical propositional logic and justification logic.

5.1 Probabilistic Logics over Classical Propositional Logic

Let CP denote classical propositional logic. If we define as basic formulas the atomic propositions (i.e. elements of the set Prop) and if we define as evaluations traditional truth assignments for classical propositional logic (on the set Prop), then we can define the non-iterated and the iterated probabilistic logic over classical propositional logic, which according to our notation are PCP and PPCP respectively. These logics have already been defined in [26] as LPP$_2$ and LPP$_1$ respectively. So, we have have the following corollary:

Corollary 22.

1. LPP$_2$SAT is NP-complete.

2. LPP$_1$SAT is PSPACE-complete.
Proof. The satisfiability problem for $\text{cpnb}$-formulas in classical propositional logic can be decided in polynomial time (we simply have to check whether the $\text{cpnb}$ formula contains an atomic proposition and its negation). So if we set $C = P$ in Theorems 14 and 19 we conclude that $\text{LPP}_{2, \text{SAT}} \in \text{NP}$ and that $\text{LPP}_{1, \text{SAT}} \in \text{PSPACE}$. The lower bounds follow from Theorems 21 and 15.

5.2 Probabilistic Logics over Justification Logic

Before defining the non-iterated and the iterated probabilistic justification logic, we briefly recall justification logic $J$ and its satisfiability algorithm.

5.2.1 The basic Justification Logic $J$

The language of justification logic [2] is defined by extending the language of classical propositional logic with formulas of the form $t : \alpha$. In the formula $t : \alpha$, $t$ is a justification term, which is used to represent evidence, and $\alpha$ is a justification formula, which is used to represent propositions, statements or facts. As we will see later, formula $\alpha$ might contain terms as well. The formula $t : \alpha$ reads as "$t$ is a justification for believing $\alpha$" or as "$t$ justifies $\alpha$". For example, assume that we have an agent who sees a snake behind him/her. Whereas in traditional modal logic we can express a statement like "the agent believes/knows that he/she is in danger", in justification logic we can express a statement like "the agent is in danger because there is a snake behind him/her". In the last statement an observation of the snake can serve as a justification. So, in justification logic the representation of knowledge becomes explicit.

Justification terms are built from countably many constants and countably many variables according to the following grammar:

$$t ::= c \mid x \mid (t \cdot t) \mid (t + t) \mid !t$$

where $c$ is a constant and $x$ is a variable. $\text{Tm}$ denotes the set of all terms and $\text{Con}$ denotes the set of all constants. For $c \in \text{Con}$ and $n \in \mathbb{N}$ we define:

$$!^0c := c \quad \text{and} \quad !^{n+1}c := !(!^nc).$$

The operators $\cdot$ and $+$ are assumed to be left-associative. The intended meaning of the connectives used in the set $\text{Tm}$ will be clear when we present the deductive system for $J$.

Formulas of the language $\mathcal{L}_J$ (justification formulas) are built according to the following grammar:

$$\alpha ::= p \mid t : \alpha \mid \neg \alpha \mid \alpha \land \alpha$$

where $t \in \text{Tm}$ and $p \in \text{Prop}$. Following our previous notation we have:

$$\mathcal{B}(\mathcal{L}_J) = \text{Prop} \cup \{t : \alpha \mid \alpha \in \mathcal{L}_J\}.$$  

The deductive system for $J$ is the Hilbert system presented in Table 2. Axiom (J) is also called the application axiom and is the justification logic analogue of
Axioms:

(P) finite set of axiom schemata axiomatizing classical propositional logic in the language $L_J$

(J) \( \vdash s : (\alpha \to \beta) \to (t : \alpha \to s \cdot t : \beta) \)

(+) \( \vdash (s : \alpha \lor t : \alpha) \to s + t : \alpha \)

Rules:

(MP) if \( T \vdash \alpha \) and \( T \vdash \alpha \to \beta \) then \( T \vdash \beta \)

(AN!) \( \vdash !^n c : !^{n-1} c : \cdots : ! c : c : \alpha \), where \( c \) is a constant, \( \alpha \) is an instance of (P), (J) or (+) and \( n \in \mathbb{N} \)

Table 2: The Deductive System $J$

application axiom in modal logic. It states that we can combine a justification for \( \alpha \to \beta \) and a justification for \( \alpha \) in order to obtain a justification for \( \beta \). Axiom (+), which is also called the monotonicity axiom, states that if \( s \) or \( t \) is a justification for \( \alpha \) then the term \( s + t \) is also a justification for \( \alpha \). Rule (AN!) states that any constant can be used to justify any axiom and also that we can use the operator ! to express positive introspection: if \( c \) justifies axiom instance \( \alpha \), then \( ! c \) justifies \( c : \alpha \), \( !! c \) justifies \( c : c : \alpha \) and so on. In justification logic it is common to assume that only some constants justify some axioms (see the notion of constant specification in [2]). However, for the purposes of this paper it suffices to assume that every constant justifies every axiom (this assumption corresponds to the notion of a total constant specification [2]).

In order to illustrate the usage of axioms and rules in $J$ we present the following example:

Example 23. Let \( a, b \in \text{Con} \), \( \alpha, \beta \in L_J \) and \( x, y \) be variables. Then we have the following:

\[ \vdash_J (x : \alpha \lor y : \beta) \to a \cdot x + b \cdot y : (\alpha \lor \beta) \]

Proof. Since \( \alpha \to \alpha \lor \beta \) and \( \beta \to \alpha \lor \beta \) are instances of (P), we can use (AN!) to obtain:

\[ \vdash_J a : (\alpha \to \alpha \lor \beta) \]

and:

\[ \vdash_J b : (\beta \to \alpha \lor \beta). \]

Using (J) and (MP) we obtain:

\[ \vdash_J x : \alpha \to a \cdot x : (\alpha \lor \beta) \]

and

\[ \vdash_J y : \beta \to b \cdot y : (\alpha \lor \beta) \]

Using (+) and propositional reasoning we obtain:

\[ \vdash_J x : \alpha \to a \cdot x + b \cdot y : (\alpha \lor \beta) \]
and

\[ \vdash_J y : \beta \rightarrow a \cdot x + b \cdot y : (\alpha \lor \beta) \]

We can now obtain the desired result by applying propositional reasoning. ■

Logic J also enjoys the *internalization property*, which is presented in the following theorem. Internalization states that the logic internalizes its own notion of proof. The version without premises is an explicit form of the necessitation rule of modal logic. A proof of the following theorem can be found in [21].

**Theorem 24 (Internalization).** For any \( \alpha, \beta_1, \ldots, \beta_n \in \mathcal{L}_J \) and \( t_1, \ldots, t_n \in \text{Tm} \), if:

\[ \beta_1, \ldots, \beta_n \vdash_J \alpha \]

then there exists a term \( t \) such that:

\[ t_1 : \beta_1, \ldots, t_n : \beta_n \vdash_J t : \alpha . \]

The models for J which we are going to present are called \( \text{M-models} \) and were introduced by Mkrtychev [23] for the logic LP. Later Kuznets [19] adapted these models for other justification logics (including J) and proved the corresponding soundness and completeness theorems. Formally, we have:

**Definition 25 (M-Model).** An \( \text{M-model} \) is a pair \( \langle v, E \rangle \), where \( v : \text{Prop} \rightarrow \{T,F\} \) and \( E : \text{Tm} \rightarrow \mathcal{P}(\mathcal{L}_J) \) such that for every \( s, t \in \text{Tm} \), for \( c \in \text{Con} \) and \( \alpha, \beta \in \mathcal{L}_J \), for \( \gamma \) being an axiom instance of J and \( n \in \mathbb{N} \) we have:

1. \( (\alpha \rightarrow \beta \in E(s) \text{ and } \alpha \in E(t)) \Rightarrow \beta \in E(s \cdot t) ; \)
2. \( E(s) \cup E(t) \subseteq E(s + t) ; \)
3. \( !^{n-1}c : !^{n-2}c : \cdots : !c : c : \gamma \in E(!^{n}c) . \)

**Definition 26 (Satisfiability in an M-model).** We define what it means for an \( \mathcal{L}_J \)-formula to hold in the \( \text{M-model} \) \( M = \langle v, E \rangle \) inductively as follows (the connectives \( \neg \) and \( \land \) are treated classically):

\[ M \models p \iff v(p) = T \quad \text{for } p \in \text{Prop} ; \]
\[ M \models t : \alpha \iff \alpha \in E(t) . \]

We close this section by briefly recalling the known complexity bounds for \( J_{\text{SAT}} \). The next theorem is due to Kuznets [19, 20]. We present it here briefly using our own notation.

**Theorem 27.** The satisfiability problem for cnpb-formulas in logic J belongs to coNP.

**Proof.** Let \( a \) be the cnpb-formula of logic J that is tested for satisfiability. Assume that there is no \( p \in \text{Prop} \) such that \( p \) appears both positively and negatively in \( a \) (otherwise it is clear that \( a \) is not satisfiable). So, the satisfiability of \( a \) depends only on the justification assertions that appear in \( a \).
Let $p_i : \alpha_i$ be the assertions that appear positively in $a$ and let $n_i : \beta_i$ be the assertions that appear negatively in $a$. If we can show that the question “does a model, which satisfies all the $p_i : \alpha_i$’s and falsifies all the $n_i : \beta_i$’s, exist?” has a short (i.e. of polynomial size and polynomially verifiable with respect to $|a|$) no-certificate, we are done. For this purpose it suffices to guess some $n_j : \beta_j$ and show that every $M$-model that satisfies all the $p_i : \alpha_i$’s, satisfies $n_j : \beta_j$ too. At first, it seems impossible to verify that our guess is correct in finite time, since we have that some terms justify infinitely many formulas (in particular every constant justifies all the axiom instances, which are infinitely many). However, since $J$ is axiomatized by finitely many axiom schemes we can use schematic variables for formulas and terms. This way we have that every constant justifies only finitely many axiom schemes. So, the short no-certificate can be guessed as follows: we non-deterministically choose some $n_j : \beta_j$. The term $n_j$ is created by the connectives $\cdot$ and $+$ using a finite tree like the following one:

```
   n_j
  /   \
 /     \
/      /
\      \n\      \n\      \n\      \n\      \n\      \n\      \n```

The above tree can be constructed in many ways but we guess one, such that in the leaves there are terms of the form $!^n c : \cdots : !c : c$ or some $p_i$’s (if we cannot make this guess then $n_j$ does not justify $\beta_j$ and our initial guess is incorrect). We assign to each of these leaves an axiom scheme (if the leaf is of the form $!^n c : \cdots : !c : c$) or an $\alpha_i$ (if the leaf is of the form $p_i$). Then we verify, using unification, that we can reach $\beta_j$ in the root (whenever we have a node of the form $+$, we make a non deterministic choice). The formulas that we assigned to the leaves and the structure of the tree compose the short certificate and it can be shown that representing formulas as directed acyclic graphs and using Robinson’s unification algorithm [8] we can verify that the unifications succeed in polynomial time. Hence the satisfiability for $\text{cpnb}$-formulas in logic $J$ belongs to $\text{coNP}$. ■

Finally, we can present the known complexity bounds for $J_{\text{SAT}}$.

**Theorem 28.** $J_{\text{SAT}}$ is $\Sigma^p_2$-complete.

*Proof. Let $\alpha \in L_j$ be the formula that is tested for satisfiability. The upper bound follows by Kuznets’s algorithm [19] which can be described by the following steps, using our notation:

1. Create a node with $w \top \alpha$. Apply the propositional rules for as long as possible. Non-deterministically choose a world path $p_w$. This can be done in non deterministic polynomial time.
2. Verify that $P_{p_w}$ is satisfiable using the coNP-algorithm of Theorem 27.

Observe that there is no rule that produces new worlds in the above algorithm. We have presented the algorithm using world signs in order to be consistent with our tableau notation. The lower bound follows by a result of Achilleos [1].

5.2.2 Probabilistic Justification Logic

Let $M$ be an $M$-model. Based on $M$ we can define the evaluation $v_M$ as follows:

$$\text{for every } \beta \in B(L_J), \ v_M(\beta) = T \iff M \models \beta.$$ 

So, if we set $L = J$, $L = L_J$ and we define the evaluations as above, we can define the non-iterated probabilistic logic over $J$, which is the logic $PJ$. For the complexity of the satisfiability problem in $PJ$, we have the following corollary:

**Corollary 29.** $PJ_{SAT}$ is $\Sigma^p_2$-complete.

**Proof.** The upper bound follows from Theorems 27 and 14. We can prove the lower bound via a reduction from $J_{SAT}$ using the same argument as in Theorem 15.

Now we will present the iterated probabilistic justification logic $PPJ$. The language of $PPJ$ is called $L_{PPJ}$ and is defined by the following grammar:

$$A ::= p \mid t : A \mid \neg A \mid A \land A \mid P_{\geq s}A$$

where $p \in \text{Prop}$ and $s \in \mathbb{Q} \cap [0,1]$. According to our previous definitions we have $B(L_{PPJ}) = \text{Prop} \cup \{t : A \mid A \in L_{PPJ}\}$. In order to define the $PPJ$-models we have to define evaluations for basic $PPJ$-formulas. Since basic $PPJ$-formulas resemble basic $J$-formulas, it makes sense to extend the definition of $M$-models in order to define evaluations for $PPJ$. So, as in the case of logic $J$ we have to present the deductive system of $PPJ$ first. This system is presented in Table 3. The axiomatization of $PPJ$ is a combination of the axiomatization for $LPP_1$ [26] and of the axiomatization for the basic justification logic $J$. Axiom (NN) corresponds to the fact that the probability of truthfulness of every formula is at least 0 (the acronym (NN) stands for non-negative). Observe that by substituting $\neg A$ for $A$ in (NN), we have $P_{\geq 0} \neg A$, which by our syntactical abbreviations is $P_{\geq 1}A$. Hence axiom (NN) also corresponds to the fact that the probability of truthfulness for every formula is at most 1. Axioms (L1) and (L2) describe some properties of inequalities (the L in (L1) and (L2) stands for less). Axioms (Add1) and (Add2) correspond to the additivity of probabilities for disjoint events (the Add in (Add1) and (Add2) stands for additivity). Rule (PN) is the probabilistic analogue of the necessitation rule in modal logics (hence the acronym (PN) stands for probabilist necessitation): if a formula is a validity, then it has probability 1. Rule (ST) intuitively states that if the probability of a formula is arbitrary close to $s$, then it is at least $s$. Observe that the rule (ST) is infinitary in the sense that it has an infinite number of premises. It corresponds to the Archimedean property for the real numbers. The acronym (ST)
Axioms:

(P) finitely many axiom schemata axiomatizing classical propositional logic in the language \( L \)

(NN) \( \vdash P_{\geq 0}A \)
(L1) \( \vdash P_{\leq r}A \rightarrow P_{< s}A \), where \( s > r \)
(L2) \( \vdash P_{< s}A \rightarrow P_{\leq r}A \)
(Add1) \( \vdash P_{\geq r}A \land P_{\geq s}B \land P_{\geq 1} \neg (A \land B) \rightarrow P_{\geq \min(1, r+s)}(A \lor B) \)
(Add2) \( \vdash P_{\geq r}A \land P_{< s}B \rightarrow P_{< r+s}(A \lor B) \), where \( r + s \leq 1 \)
(J) \( \vdash s : (A \rightarrow B) \rightarrow (t : A \rightarrow s \cdot t : B) \)
(+) \( \vdash (s : A \lor t : A) \rightarrow s + t : A \)

Rules:

(MP) if \( T \vdash A \) and \( T \vdash A \rightarrow B \) then \( T \vdash B \)
(PN) if \( \vdash A \) then \( \vdash P_{\geq 1}A \)
(ST) if \( T \vdash A \rightarrow P_{\geq k}B \) for every integer \( k \geq \frac{1}{s} \) and \( s > 0 \) then \( T \vdash A \rightarrow P_{\geq s}B \)
_AN!_ \( \vdash !^nc : !^{n-1}c : \cdots : !c : c : A \), where \( c \in \text{Con} \), \( A \) is an instance of some PPJ-axiom and \( n \in \mathbb{N} \)

Table 3: The Deductive System PPJ

stands for strengthening, since the statement of the result is stronger than the statement of the premises. Rule (ST) was introduced in [25, 29] so that strong completeness for probabilistic logics could be proved. We recall that a logical system is strongly complete if and only if every consistent set (finite or infinite) has a model. As it is shown in [25, 29], languages used for probabilistic logics are non-compact, so the proof of strong completeness is impossible without an infinitary rule. Observe that in PPJ constants instances of probabilistic axioms (not only justification axioms as in J).

So, now we can extend Definition 25 to the basic formulas of \( L_{\text{PPJ}} \).

**Definition 30** (Extended M-Model). An extended M-model is a pair \( (v, \mathcal{E}) \), where \( v : \text{Prop} \rightarrow \{T, F\} \) and \( \mathcal{E} : \text{Tm} \rightarrow \mathcal{P}(\mathcal{L}_{\text{PPJ}}) \) such that for every \( s, t \in \text{Tm} \), for \( c \in \text{Con} \) and \( A, B \in \mathcal{L}_{\text{PPJ}} \), for \( C \) being an axiom instance of PPJ and \( n \in \mathbb{N} \) we have:

1. \( (A \rightarrow B \in \mathcal{E}(s) \text{ and } A \in \mathcal{E}(t)) \implies B \in \mathcal{E}(s \cdot t) \);
2. \( \mathcal{E}(s) \cup \mathcal{E}(t) \subseteq \mathcal{E}(s + t) \);
3. \( !^nc : !^{n-1}c : \cdots : !c : c : C \in \mathcal{E}(!^n c) \).

We can define evaluations based on extended M-models in the same way as for the standard M-models. So a PPJ-model is a PPL-model where the
evaluations are based on extended M-models. This completes the definition of semantics for PPJ.

Now we are ready to present the complexity bounds for PPJ_{SAT}.

**Theorem 31.** The satisfiability problem for cpnb-formulas in logic PPJ belongs to coNP.

**Proof.** We recall that cpnb-formulas in L_{PPJ} are conjunctions of positive and negative atomic propositions and positive and negative formulas of the form \( t : A \), where \( A \in L_{PPJ} \).

Since the cpnb-formulas in L_{PPJ} resemble the cpnb-formulas in L_{J} we will use a slight variation of the algorithm in Theorem 27 for deciding the satisfiability of cpnb-formulas. The algorithm of Theorem 27 does not depend on what the axioms are, as long as they are finitely many. Assume that \( a \) is the cpnb-formula of logic PPJ that we want to test for satisfiability. As in the proof of Theorem 27 we can a formula \( t : B \) that appears negatively in \( a \), guess a tree that describes the construction of \( t \), assign to the leaves of tree PPJ-axioms and formulas and then verify that our guess is correct. However, since the axioms of PPJ have different form than the axioms of J we have to modify the verification process. Simple unification is not sufficient any more, since the axioms of PPJ come with linear conditions. In the rest of the proof we explain that the verification can be done in polynomial time by using a unification algorithm and by testing a linear system for satisfiability.

For L_{PPJ} we need three kinds of schematic variables: for terms, formulas and rational numbers. Also, because of the side conditions that come with the axioms (L1) and (Add2) our schematic formulas should be paired with systems of linear inequalities. For example, the scheme (L1) should be represented by the schematic formula \( P \leq r A \rightarrow P < s A \) (with the schematic variables \( r \), \( s \), and \( A \)) together with the inequality \( r < s \), whereas a scheme that is obtained by a conjunction of the schemata (L1) and (Add2) should be represented as

\[
(P \leq r_1 A_1 \rightarrow P < s_1 A_1) \land (P \leq r_2 A_2 \land P < s_2 B_2 \rightarrow P < r_2 + s_2 (A_2 \lor B_2))
\]

together with the inequalities

\[
\{ r_1 < s_1, r_2 + s_2 \leq 1 \}.
\]

We should not forget that the rational variables belong to \( \mathbb{Q} \cap [0, 1] \). So we have to add constraints like \( 0 \leq r \leq 1 \). Hence in addition to constructing unification equations we need to take care of the linear constraints. For instance, in order to unify the schemata \( P_{\geq r} A \) and \( P_{\geq s} B \) the algorithm has to unify \( A \) and \( B \), and to equate \( r \) and \( s \), i.e. it adds \( r = s \) to the linear system. At the end the verification algorithm will succeed only if the standard unification of formulas succeeds and the linear system is solvable.

Another complication are constraints of the form

\[
l = \min(1, r + s)
\]
that originate from the scheme (Add1). Obviously, Eq. (16) is not linear. However, we find that Eq. (16) has a solution if and only if one of the set of equations
\[ \{ l = r + s, r + s \leq 1 \} \text{ or } \{ l = 1, r + s > 1 \} \]
has a solution. Thus whenever we come to an equation like (16) we can nondeterministically chose one of the equivalent set of equations and add it to the constructed linear system.

We conclude that we can guess a tree for t and also a linear system in nondeterministic polynomial time. We also find that the verification can be done in polynomial time, since testing a linear system for satisfiability can be done in polynomial time [13] and unification of formulas can be checked in polynomial time using Robinson’s algorithm. So, as in the case of J we can show that the satisfiability problem for cpnb-formulas in PPJ has short no-certificates, i.e it belongs to coNP. ■

**Corollary 32.** PPJ\textsc{Sat} is \textsc{PSPACE}-complete.

**Proof.** The upper bound follows from Theorems 31 and 19. The lower bound follows from Theorem 21. ■

### 6 Conclusion

We have presented upper and lower complexity bounds for the satisfiability problem in non-iterated and iterated probabilistic logics over any extension of classical propositional logic. The aforementioned bounds are parameterized on the complexity of satisfiability of conjunctions of positive and negative formulas that have neither a probabilistic nor a classical operator as their top-connectives. As an application we have shown how tight bounds for the complexity of satisfiability in non-iterated and iterated probabilistic logics over classical propositional logic and justification logic can be obtained. It is interesting that both for classical propositional logic and for the basic justification logic J adding non-nested probabilistic operators to the language does not increase the complexity of the satisfiability problem.

An interesting open problem that is related to the research in this paper is the following: Rašković et al. [30] define a probabilistic logic over classical propositional logic, called LPP\textsuperscript{S}, using approximate conditional probabilities with the intention to model non-monotonic reasoning. The satisfiability problem for LPP\textsuperscript{S} is again reduced to solving linear systems. In [28] the logic of [30] is extended to a probabilistic logic with approximate conditional probabilities over justification logic. What are the complexity bounds for the satisfiability problem in the logics of [30] and [28]?

### References

[1] A. Achilleos. Nexp-completeness and universal hardness results for justification logic. In L. D. Beklemishev and D. V. Musatov, editors, *Computer*
[2] S. Artemov and M. Fitting. Justification logic. In E. N. Zalta, editor, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016.

[3] S. N. Artemov. Operational modal logic. Technical Report MSI 95–29, Cornell University, Dec. 1995.

[4] S. N. Artemov. Explicit provability and constructive semantics. Bulletin of Symbolic Logic, 7(1):1–36, 2001.

[5] P. Blackburn, M. de Rijke, and Y. Venema. Modal Logic. Cambridge University Press, New York, NY, USA, 2001.

[6] V. N. Brezhnev. On explicit counterparts of modal logics. Technical Report CFIS 2000–05, Cornell University, 2000.

[7] V. Chvátal. Linear programming. W. H. Freeman and Company, New York, 1983.

[8] J. Corbin and M. Bidoit. A Rehabilitation of Robinson’s Unification Algorithm. Comptes Rendus de l’ Académie des Sciences. Serie I-Mathematique, 296(5):279–282, 1983.

[9] R. Fagin, J. Halpern, and N. Megiddo. A logic for reasoning about probabilities. Information and Computation, 87:78–128, 1990.

[10] R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. Journal of the ACM (JACM), 41(2):340–367, 1994.

[11] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. Journal of complexity, 4(1):1–11, 1988.

[12] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. Artif. Intell., 54(2):319–379, 1992.

[13] N. Karmarkar. A new polynomial-time algorithm for linear programming. Combinatorica, 4(4):373–395, Dec 1984.

[14] J. Keisler. Hyperfinite model theory. In R. O. Gandy and J. M. E. Hyland, editors, Logic Colloquium 1976, Lecture Notes in Computer Science, page 5–10. North-Holland, 1977.

[15] I. Kokkinis. The complexity of non-iterated probabilistic justification logic. In M. Gyssens and G. R. Simari, editors, Foundations of Information and Knowledge Systems - 9th International Symposium, FoIKS 2016, Linz, Austria, March 7-11, 2016. Proceedings, volume 9616 of Lecture Notes in Computer Science, pages 292–310. Springer, 2016.
[16] I. Kokkinis. The complexity of probabilistic justification logic. CoRR, abs/1708.04100, 2017.

[17] I. Kokkinis, P. Maksimović, Z. Ognjanović, and T. Studer. First steps towards probabilistic justification logic. Logic Journal of the IGPL, 23(4):662–687, 2015.

[18] I. Kokkinis, Z. Ognjanovic, and T. Studer. Probabilistic justification logic. In S. N. Artemov and A. Nerode, editors, Logical Foundations of Computer Science - International Symposium, LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings, volume 9537 of Lecture Notes in Computer Science, pages 174–186. Springer, 2016.

[19] R. Kuznets. On the complexity of explicit modal logics. In P. G. Clote and H. Schwichtenberg, editors, Computer Science Logic, 14th International Workshop, CSL 2000, Annual Conference of the EACSL, Fischbachau, Germany, August 21–26, 2000, Proceedings, volume 1862 of Lecture Notes in Computer Science, pages 371–383. Springer, 2000.

[20] R. Kuznets. Complexity Issues in Justification Logic. PhD thesis, CUNY Graduate Center, May 2008.

[21] R. Kuznets and T. Studer. Justifications, ontology, and conservativity. In T. Bolander, T. Braüner, S. Ghilardi, and L. Moss, editors, Advances in Modal Logic, Volume 9, pages 437–458. College Publications, 2012.

[22] H. E. J. Kyburg. Probability and the Logic of Rational Belief. Wesleyan University Press, 1961.

[23] A. Mkrtychev. Models for the logic of proofs. In S. Adian and A. Nerode, editors, Logical Foundations of Computer Science, 4th International Symposium, LFCS'97, Yaroslavl, Russia, July 6–12, 1997, Proceedings, volume 1234 of Lecture Notes in Computer Science, pages 266–275. Springer, 1997.

[24] N. Nilsson. Probabilistic logic. Artificial Intelligence, 28:71–87, 1986.

[25] Z. Ognjanović and M. Rašković. Some first-order probability logics. Theoretical Computer Science, 247(1):191–212, 2000.

[26] Z. Ognjanović, M. Rašković, and Z. Marković. Probability logics. Zbornik radova, subseries “Logic in Computer Science”, 12(20):35–111, 2009.

[27] Z. Ognjanović, M. Rašković, and Z. Marković. Probability Logics - Probability-Based Formalization of Uncertain Reasoning. Springer, 2016.

[28] Z. Ognjanović, N. Savić, and T. Studer. Justification logic with approximate conditional probabilities. In A. Baltag, J. Seligman, and T. Yamada, editors, Logic, Rationality, and Interaction - 6th International Workshop, LORI 2017, Sapporo, Japan, September 11-14, 2017, Proceedings, volume 10455 of Lecture Notes in Computer Science, pages 681–686. Springer, 2017.
[29] M. Rašković and Z. Ognjanović. A first order probability logic–$L_{PQ}$. *Publications de l’Institut Mathématique. Nouvelle Série*, 65:1–7, 1999.

[30] M. Rašković, Z. Marković, and Z. Ognjanović. A logic with approximate conditional probabilities that can model default reasoning. *Int. J. Approx. Reasoning*, 49(1):52–66, 2008.