Satisfiability for Probability Modalities

Marcelo Finger[1], Tommaso Flaminio[2], Lluís Godo[2], Felip Manyà[2] and Sandro Preto[3]

[1]University of São Paulo, São Paulo, Brazil
[2]Artificial Intelligence Research Institute, Spanish National Research Council, Bellaterra, Spain
[3]Federal University of ABC, Santo André, Brazil

Abstract

The fuzzy logical system FP(Ł) extends Łukasiewicz logic by replacing propositional variables with basic modal formulas of the form $P\varphi$, expressing the fuzzy notion that a classical event φ is probable. In this work, we study the satisfiability problem for FP(Ł) by presenting an algorithm and conducting an empirical evaluation over a controlled class of FP(Ł)-satisfiability instances. Our experimental results reveal a clear phase transition behaviour and distinctive running-time patterns, shedding light on the computational properties of FP(Ł)-satisfiability.

Keywords

Probability, Łukasiewicz logic, satisfiability

1. Introduction

Fuzzy and probability logics are traditionally seen as distinct reasoning systems. On the one hand, fuzzy logic semantics deal with degrees of truth, generalizing the two truth values of classical logic. On the other hand, probability logic semantics are concerned with degrees of certainty about events expressed by logical formulas, see e.g. [1]. Nevertheless, there are proposals that identify both approaches by regarding the probability value of an event φ as the fuzzy truth value of $P\varphi$, meaning "it is probable that φ". In this way, being probable is considered a fuzzy concept, and P is a unary modal operator in the object language.

The logical system FP(Ł) is one such proposal [2, 3].[1] It is built upon Łukasiewicz logic Ł, a well-established and extensively studied system of propositional fuzzy logic [5], together with classical probability logic. Formulas of FP(Ł) are constructed in the same way as formulas of Ł; however, instead of propositional variables, they use *basic modal formulas* of the form $P\varphi$, where φ is a classical propositional formula and P is the probability modal operator. For example, while $x \oplus \neg y$ is a Ł-formula with Ł-propositional variables x and y, the expression $P(\neg X \vee Y) \oplus \neg P(X \wedge Y)$ is an FP(Ł)-formula, where $\neg X \vee Y$ and $X \wedge Y$ are classical propositional formulas over the classical propositional variables X and Y.

Among its main properties, FP(Ł) is sound and complete with respect to probability models [3]. The functional representation of FP(Ł)-formulas has been studied [6]. In addition, the satisfiability problem for FP(Ł)-formulas has been shown to be **NP**-complete [7].

Although the **NP**-completeness of the satisfiability problem for FP(Ł) has been established via a nondeterministic argument [7], an explicit algorithmic procedure suitable for implementation has not yet been described in the literature. In this work, we propose such an algorithm in full detail, based on a reduction from the satisfiability problem to a mixed-integer linear programming (MILP) instance. The computation proceeds through a branching procedure combined with the multiple solution of linear problems, where a column generation method is employed.

Practical Aspects of Automated Reasoning (PAAR) 2026, as part of the Federated Logic Conference (FLoC) 2026, Lisbon, Portugal, July 25, 2026

mfinger@ime.usp.br (M. Finger); tommaso@iiia.csic.es (T. Flaminio); godo@iiia.csic.es (L. Godo); felip@iiia.csic.es (F. Manyà); sandro.preto@ufabc.edu.br (S. Preto)
0000-0002-1391-1175 (M. Finger); 0000-0002-9180-7808 (T. Flaminio); 0000-0002-6929-3126 (L. Godo); 0000-0002-8366-1458 (F. Manyà); 0000-0003-4448-5364 (S. Preto)

[1]Despite the approaches being quite different, in [4] the authors have defined mutual translations between the *classical* probability logic of [1] and FP(Ł).

We also analyze the empirical behavior of this algorithm through experiments using a publicly available implementation, revealing a phase transition phenomenon and well-defined running-time patterns for a parameterized class of FP(Ł)-formula instances.

The rest of this paper is organized as follows. Section 2 introduces propositional classical and Łukasiewicz logics, probabilities over classical formulas, and the system FP(Ł). Section 3 presents the linear formulation of the semantics and the satisfiability problem for FP(Ł), while Section 4 establishes a decision procedure for this problem. Section 5 reports empirical results from an implementation of the proposed algorithm. Finally, Section 6 concludes the paper.

2. Classical and Łukasiewicz Logics, Probabilities, and a Combination of Them All

Classical propositional logic (CPL) is based on a propositional language $\mathcal{L}_{\text{CPL}}$ freely generated from a countable set of propositional variables $\{X_1, X_2, \ldots\}$ through the binary CPL-disjunction operator $\vee$ and the unary CPL-negation operator $\neg$. We denote CPL-formulas (in $\mathcal{L}_{\text{CPL}}$) by lower Greek letters. Other operators as $\wedge$, $\rightarrow$ and $\leftrightarrow$ may be derived from $\vee$ and $\neg$ as usual. Also, the semantics of CPL is defined as usual through CPL-valuations over $\mathcal{L}_{\text{CPL}}$. The two classical truth values are taken to be 0 and 1 throughout this text. We say that a CPL-formula φ is a *tautology* if $v(\varphi) = 1$, for all CPL-valuation v.

A *probability function* over $\mathcal{L}_{\text{CPL}}$ is a function $\mu : \mathcal{L}_{\text{CPL}} \rightarrow [0, 1]$ such that, for any CPL-formulas $\varphi, \psi \in \mathcal{L}_{\text{CPL}}$:

- If φ is a tautology, then $\mu(\varphi) = 1$; and
- If $\neg(\varphi \wedge \psi)$ is a tautology, then $\mu(\varphi \vee \psi) = \mu(\varphi) + \mu(\psi)$.

The *classical probabilistic satisfiability problem*, which we refer to as Psat, asks whether there is a probability function μ that agrees with given probability values assigned to a finite amount of CPL-formulas. If such a probability function exists, we say that the probability values assignment is *coherent*, otherwise, it is *incoherent*. The Psat problem is **NP**-complete [8].[2]

A *partial CPL-valuation* is an assignment of classical truth values to a finite set of propositional variables $V = \{X_1, \ldots, X_n\}$. Note that a partial CPL-valuation over V may be extended to infinitely many CPL-valuations that agree on all CPL-formulas built from the propositional variables in V. A *probability distribution* is an assignment of probability values (so in $[0, 1]$) to all the partial CPL-valuations over a finite set of propositional variables that sums up to 1. Let a Psat-instance be given by pairs $\langle \varphi_1, b_1 \rangle, \ldots, \langle \varphi_k, b_k \rangle$, where each φ_i is a CPL-formula and $b_i \in [0, 1]$ its assignment, such that the propositional variables occurring in $\varphi_1, \ldots, \varphi_k$ are those in set V. Such Psat-instance is coherent if, and only if, there is a probability distribution m over the partial CPL-valuations such that

$$b_i = \sum_{j=1}^{2^n} m(c_j) \cdot c_j(\varphi_i), \tag{1}$$

for all $i \in \{1, \ldots, k\}$, where the c_j's are the 2^n partial CPL-valuations over V.

Łukasiewicz logic (Ł) is based on a propositional language $\mathcal{L}_{\text{Ł}}$ freely generated from a countable set of propositional variables through the binary Ł-disjunction operator $\oplus$ and the unary Ł-negation operator $\neg$ (there will be no risk of confusion with the CPL-negation symbol). We denote Ł-formulas (in $\mathcal{L}_{\text{Ł}}$) by capital Greek letters. For the semantics of Ł, a Ł-valuation is a function $v : \mathcal{L}_{\text{Ł}} \rightarrow [0, 1]$ that obeys, for $\Phi, \Psi \in \mathcal{L}_{\text{Ł}}$:

$$v(\neg\Phi) = 1 - v(\Phi); \text{ and} \tag{2}$$

$$v(\Phi \oplus \Psi) = \min(1, v(\Phi) + v(\Psi)). \tag{3}$$

From Ł-disjunction and Ł-negation one may derive the Ł-operators in Table 1.

[2] Strictly speaking, the Psat problem, when studied from a computational perspective, considers only rational probability assignments. For the purposes of this work, however, such a restriction is not required.

Table 1
Derived operators in Ł

Operator	Ł-valuation
Conjunction: $\Phi \odot \Psi =_{\text{def}} \neg(\neg\Phi \oplus \neg\Psi)$	$v(\Phi \odot \Psi) = \max(0, v(\Phi) + v(\Psi) - 1)$
Maximum (weak disjunction): $\Phi \vee \Psi =_{\text{def}} \neg(\neg\Phi \oplus \Psi) \oplus \Psi$	$v(\Phi \vee \Psi) = \max(v(\Phi), v(\Psi))$
Minimum (weak conjunction): $\Phi \wedge \Psi =_{\text{def}} \neg(\neg\Phi \vee \neg\Psi)$	$v(\Phi \wedge \Psi) = \min(v(\Phi), v(\Psi))$
Implication: $\Phi \to \Psi =_{\text{def}} \neg\Phi \oplus \Psi$	$v(\Phi \to \Psi) = \min(1, 1 - v(\Phi) + v(\Psi))$
Bi-implication: $\Phi \leftrightarrow \Psi =_{\text{def}} (\Phi \to \Psi) \wedge (\Psi \to \Phi)$	$v(\Phi \leftrightarrow \Psi) = 1 - \lvert v(\Phi) - v(\Psi)\rvert$

Now we define logic FP(Ł) whose operators are those of classical and Łukasiewicz logics plus a unary modality operator P. There are two kinds of formulas in FP(Ł):

- **Non-modal FP(Ł)-formulas.** These are the exact same formulas of classical propositional logic CPL, denoted by lower case Greek letters.
- **(Modal) FP(Ł)-formulas**. These are built from *basic modal formulas* of the form $P\varphi$, where φ is a non-modal FP(Ł)-formula, using the operators of Łukasiewicz logic Ł. These formulas are denoted by capital Greek letters, as Ł-formulas, with no danger of confusion. The intended meaning of the modality $P\varphi$ is "formula φ is Ł-valuated with its classical probability". Examples of FP(Ł)-formulas include $P\varphi \to P\psi \oplus P\varphi$. Note that neither nested formulas as $P(P\varphi \oplus P\psi)$, nor mixed formulas as $\varphi \to P\psi$ are allowed.[3]

In this work, we focus on modal FP(Ł)-formulas, which we simply refer to as FP(Ł)-formulas. We say that the subformulas of an FP(Ł)-formula Φ are all the FP(Ł)-formulas appearing in its inductive construction, starting from its basic modal formulas; so Φ itself is a subformula of Φ. The set of FP(Ł)-formulas is denoted by $\mathcal{L}_{\text{FP(Ł)}}$.

For the semantics of FP(Ł)-formulas, first let $\mu : \mathcal{L}_{\text{CPL}} \to [0, 1]$ be a probability function over $\mathcal{L}_{\text{CPL}}$. Then, an FP(Ł)-valuation based on μ is a function $v : \mathcal{L}_{\text{FP(Ł)}} \to [0, 1]$ that obeys:

- Equation $v(P\varphi) = \mu(\varphi)$, for any basic modal formula $P\varphi \in \mathcal{L}_{\text{FP(Ł)}}$; and
- Equations (2)–(3), seeing Φ and Ψ as FP(Ł)-formulas $\Phi, \Psi \in \mathcal{L}_{\text{FP(Ł)}}$.

Note that a Ł-valuation is not in general an FP(Ł)-valuation since it does not necessarily assign probabilistically coherent values to basic atomic formulas.

The main goal of this work is to propose a solving procedure to decide the *satisfiability problem of FP(Ł)-formulas*, which we refer to as the FP(Ł)-SAT problem: given an FP(Ł)-formula Φ, is there an FP(Ł)-valuation v such that $v(\Phi) = 1$? If such a valuation exists, we say that Φ is *satisfiable*, otherwise, it is *unsatisfiable*. For instance, $P\psi \to P\varphi$ is satisfiable if there exists a probability for which φ is at least as probable as ψ. Hájek and Tulipani showed that FP(Ł)-SAT is **NP**-complete [7]. The algorithm proposed in this work is a deterministic development of the nondeterministic procedure they outlined in their proof.

Notation. Throughout this work, we write $\Phi[P\psi_1, \ldots, P\psi_k]$ to highlight the basic modal formulas occurring in an FP(Ł)-formula Φ. The cardinality of a set B is denoted by $|B|$. Also, dotted lines in matrices are used to visually indicate their construction by concatenation of submatrices.

[3]The fact that modalities do not nest points to the fact that, in FP(Ł), probabilistic modalities are *externally applied* to Ł, in the sense of [9].

Table 2
MILP constraints for Ł-operators over modal formulas Φ and Ψ

Negation $\neg\Phi$	$x_{\neg\Phi} = 1 - x_\Phi$
Disjunction $\Phi \oplus \Psi$	$b_{\Phi\oplus\Psi} \in \{0,1\}$ $b_{\Phi\oplus\Psi} \leq x_{\Phi\oplus\Psi} \leq 1$ $x_\Phi + x_\Psi - b_{\Phi\oplus\Psi} \leq x_{\Phi\oplus\Psi} \leq x_\Phi + x_\Psi$
Conjunction $\Phi \odot \Psi$	$b_{\Phi\odot\Psi} \in \{0,1\}$ $0 \leq x_{\Phi\odot\Psi} \leq b_{\Phi\odot\Psi}$ $x_\Phi + x_\Psi - 1 \leq x_{\Phi\odot\Psi} \leq x_\Phi + x_\Psi - b_{\Phi\odot\Psi}$
Maximum $\Phi \vee \Psi$	$b_{\Phi\vee\Psi} \in \{0,1\}$ $x_\Phi \leq x_{\Phi\vee\Psi} \leq x_\Phi + b_{\Phi\vee\Psi}$ $x_\Psi \leq x_{\Phi\vee\Psi} \leq x_\Psi + 1 - b_{\Phi\vee\Psi}$
Minimum $\Phi \wedge \Psi$	$b_{\Phi\wedge\Psi} \in \{0,1\}$ $x_\Phi - b_{\Phi\wedge\Psi} \leq x_{\Phi\wedge\Psi} \leq x_\Phi$ $x_\Psi - (1 - b_{\Phi\wedge\Psi}) \leq x_{\Phi\wedge\Psi} \leq x_\Psi$
Implication $\Phi \to \Psi$	$b_{\Phi\to\Psi} \in \{0,1\}$ $b_{\Phi\to\Psi} \leq x_{\Phi\to\Psi} \leq 1$ $1 - x_\Phi + x_\Psi - b_{\Phi\to\Psi} \leq x_{\Phi\to\Psi} \leq 1 - x_\Phi + x_\Psi$
Bi-implication $\Phi \leftrightarrow \Psi$	$b_{\Phi\leftrightarrow\Psi} \in \{0,1\}$ $1 - x_\Phi + x_\Psi - 2b_{\Phi\leftrightarrow\Psi} \leq x_{\Phi\leftrightarrow\Psi} \leq 1 - x_\Phi + x_\Psi$ $-1 + x_\Phi - x_\Psi + 2b_{\Phi\leftrightarrow\Psi} \leq x_{\Phi\leftrightarrow\Psi} \leq 1 + x_\Phi - x_\Psi$

3. Linear Formulation of FP(Ł)-Sat

In this section, we show how to formulate an instance of FP(Ł)-Sat in terms of linear systems of equalities and inequalities. Let $\Phi[P\psi_1, \ldots, P\psi_k]$ be an FP(Ł)-formula.

We first encode the behavior of the Ł-operators over basic modal formulas by constraints of a mixed-integer linear programming (MILP) problem, following the general approach proposed by Hähnle [10]. For that, semantic values of the basic modal formulas $P\psi_1, \ldots, P\psi_k$ are assigned to the linear variables $x_{P\psi_1}, \ldots, x_{P\psi_k}$, respectively. Then, the semantic value of formula Φ is assigned to linear variable x_Φ as long as a set of linear constraints, denoted by $C_Ł(\Phi)$, is satisfied by such assignments. The constraints are determined by the inductive structure of formula Φ. A linear variable x_Ψ is associated to each subformula Ψ of Φ. Also, a binary variable b_Ψ is associated to each subformula Ψ of Φ that is compound through a binary Ł-operator. Each compound subformula of Φ, according to the Ł-operator in its composition, determines some of the MILP constraints in set $C_Ł(\Phi)$ as in Table 2.

In addition to the constraints imposed by the inductive structure of Φ, each variable x_Ψ associated with a subformula Ψ of Φ must correspond to a constraint "$0 \leq x_\Psi \leq 1$" in $C_Ł(\Phi)$. Then, every assignment of values to linear variables that jointly satisfies the constraints in $C_Ł(\Phi)$ yields consistent semantic values regarding the Ł-operators. That is, there is a Ł-valuation v such that $v(\Psi) = x_\Psi$, for all subformulas Ψ of Φ. Note, however, that such a Ł-valuation is not guaranteed to be an FP(Ł)-valuation since its values do not necessarily stand for coherent probabilities.

As we are interested in determining the satisfiability of Φ, we define the set of constraints $C_Ł^{\text{sat}}(\Phi)$ from $C_Ł(\Phi)$ by replacing the constraint "$x_\Phi \leq 1$", which belongs to $C_Ł(\Phi)$, with the constraint "$x_\Phi = 1$". In this way, every assignment of values to linear variables that jointly satisfy the constraints in $C_Ł^{\text{sat}}(\Phi)$ yields a Ł-valuation v such that $v(\Psi) = x_\Psi$ for all subformulas Ψ of Φ, and in particular, $v(\Phi) = 1$.

If we turn the inequalities in $C_Ł^{\text{sat}}(\Phi)$ into the *standard form* using routine tricks in linear programming, we are able to write them, along with the equality in $C_Ł^{\text{sat}}(\Phi)$, in the matrix form:

$$\begin{aligned} A_{lt}\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq 0, \end{aligned} \tag{4}$$

where $A_{lt} \in \mathbb{Z}^{L \times K}$ is a matrix of linear variable coefficients, $\mathbf{x}$ is a vector of linear variables and $\mathbf{b}$ is a vector of constant terms. Thus, variable assignments jointly satisfying (4) and the binary constraints "$b_\Psi \in \{0,1\}$" establish a Ł-valuation. Let us agree that the first k of the K columns of A_{lt} are associated to linear variables $x_{P\psi_1}, \ldots, x_{P\psi_k}$, in this order.

Example 1. *For example, the behavior of the Ł-disjunction in formula $P\varphi \oplus P\psi$ may be encoded by the following constraints in $C^{\mathrm{sat}}_{Ł}(P\varphi \oplus P\psi)$:*

$$\begin{gathered} b_{P\varphi\oplus P\psi} \in \{0,1\} \\ b_{P\varphi\oplus P\psi} \leq x_{P\varphi\oplus P\psi} = 1 \\ x_{P\varphi} + x_{P\psi} - b_{P\varphi\oplus P\psi} \leq x_{P\varphi\oplus P\psi} \leq x_{P\varphi} + x_{P\psi} \\ 0 \leq x_{P\varphi\oplus P\psi} \\ 0 \leq x_{P\varphi} \leq 1 \\ 0 \leq x_{P\psi} \leq 1 \end{gathered}$$

Inequalities above may be turned into the standard form, using auxiliary linear variables $y_1, \ldots, y_5$, yielding:

$$\begin{gathered} b_{P\varphi\oplus P\psi} + y_1 = x_{P\varphi\oplus P\psi} \\ x_{P\varphi\oplus P\psi} = 1 \\ x_{P\varphi} + x_{P\psi} - b_{P\varphi\oplus P\psi} + y_2 = x_{P\varphi\oplus P\psi} \\ x_{P\varphi\oplus P\psi} + y_3 = x_{P\varphi} + x_{P\psi} \\ x_{P\varphi} + y_4 = 1 \\ x_{P\psi} + y_5 = 1 \\ b_{P\varphi\oplus P\psi},\ x_{P\varphi\oplus P\psi},\ x_{P\varphi},\ x_{P\psi} \geq 0 \\ y_1,\ y_2,\ y_3,\ y_4,\ y_5 \geq 0 \end{gathered}$$

Then, for this example, the system of equalities in (4) has the matrix form:

$$A_{lt}\mathbf{x} = \left[\begin{array}{cccc:ccccc} 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} x_{P\varphi} \\ x_{P\psi} \\ x_{P\varphi\oplus P\psi} \\ \hdashline b_{P\varphi\oplus P\psi} \\ \hdashline y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{array}\right] = \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{array}\right] = \mathbf{b}$$

In this way, nonnegative assignments to linear variables that satisfy the matrix equation above and, in addition, the binary constraint "$b_{P\varphi\oplus P\psi} \in \{0,1\}$", encode a satisfying Ł-valuation for the FP(Ł)-formula $P\varphi \oplus P\psi$—if such assignments exist.

Now, let us establish linear constraints that encode probabilistic coherence of $P\psi_1, \ldots, P\psi_k$. Let the CPL-formulas $\psi_1, \ldots, \psi_k$ be built from the set $\{X_1, \ldots, X_n\}$ of classical propositional variables and let $c_1, \ldots, c_{2^n}$ be all the partial CPL-valuations to these propositional variables. In order to the semantic values $x_{P\psi_1}, \ldots, x_{P\psi_k}$ of the basic modal formulas stand for coherent probabilities, there must be a probability distribution over the partial CPL-valuations $c_1, \ldots, c_{2^n}$, whose probability assessments are $p_1, \ldots, p_{2^n} \in [0,1]$, respectively, such that $p_1 + \cdots + p_{2^n} = 1$ and

$$x_{P\psi_i} = \sum_{j=1}^{2^n} p_j \cdot c_j(\psi_i), \tag{5}$$

for all $i \in \{1, \ldots, k\}$.

To write the constraints (5) in matrix form, we define:

- A matrix $A'_{lb} \in \mathbb{Z}^{k\times K}$ with k rows and K columns, whose elements are defined as $a_{ij} = -1$ if $i = j$, and $a_{ij} = 0$ otherwise; and
- A matrix $A'_{rb} \in \mathbb{Z}^{k\times 2^n}$ with k rows and 2^n columns, whose elements are defined as $a_{ij} = c_j(\psi_i)$.

Then, constraints (5) may be rewritten as

$$[A'_{lb}\ \ A'_{rb}] \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} = \mathbf{0}_k, \tag{6}$$

where $\mathbf{x}$ is the vector of linear variables in (4), $\mathbf{p} = [p_1\ \cdots\ p_{2^n}]^T$ and $\mathbf{0}_k$ is the null column vector with k rows; T stands for the transposition operation. In this way, there is a vector $\mathbf{p}$ of probability assessments to $c_1, \ldots, c_{2^n}$ that satisfies (6) iff the values of $x_{P\psi_1}, \ldots, x_{P\psi_k}$ stand for coherent probabilities.

To encode the conditions for $\mathbf{p}$ to represent a probability distribution, define the matrix A_{lb} as A'_{lb} extended by appending a row of zeros at the bottom, and define the matrix A_{rb} as A'_{rb} extended by appending a row of ones at the bottom. Matrices A_{lb} and A_{rb} have $k+1$ rows. Then, rewrite (6) as

$$\begin{gathered} [A_{lb}\ \ A_{rb}] \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_k \\ 1 \end{bmatrix} \\ \mathbf{p} \geq 0 \end{gathered} \tag{7}$$

Thus, we may state that there is a vector $\mathbf{p}$ satisfying (7) iff the values of $x_{P\psi_1}, \ldots, x_{P\psi_k}$ stand for coherent probabilities.

Example 2 (Example 1 continued)**.** *Resuming the example of FP(Ł)-formula $P\varphi \oplus P\psi$, let $\varphi = X$ and $\psi = \neg X$. The system of equalities in (7) has the matrix form:*

$$[A_{lb}\ A_{rb}] \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} = \left[\begin{array}{ccccc|cc} -1 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{array}\right] \left[\begin{array}{c} x_{PX} \\ x_{P\neg X} \\ x_{PX\oplus P\neg X} \\ b_{PX\oplus P\neg X} \\ y_1 \\ \vdots \\ y_5 \\ \hline p_1 \\ p_2 \end{array}\right] = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_2 \\ 1 \end{bmatrix}.$$

Now, we are in a position to assemble the linear constraints derived in this section, in (4) and (7). For that, let $A_{rt} \in \mathbb{Z}^{L\times 2^n}$ be the null matrix with L rows and 2^n columns. An assignment of values to linear variables jointly satisfies (4) and (7) iff such an assignment satisfy

$$\begin{gathered} \begin{bmatrix} A_{lt} & A_{rt} \\ A_{lb} & A_{rb} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_k \\ 1 \end{bmatrix} \\ \mathbf{x}, \mathbf{p} \geq 0 \end{gathered} \tag{8}$$

As a consequence of the developments in this section, we are finally able to state the following result.

Theorem 1. *Let Φ be an FP(Ł)-formula and let "$b_1 \in \{0,1\}$",...,"$b_m \in \{0,1\}$" be the binary constraints in $C^{\mathrm{sat}}_{Ł}(\Phi)$. Let the matrices A_{lt}, A_{lb}, A_{rt} and A_{rb} be defined as described in this section according to Φ. Then, Φ is FP(Ł)-satisfiable if, and only if, (8) and the binary constraints "$b_1 \in \{0,1\}$",...,"$b_m \in \{0,1\}$" in $C^{\mathrm{sat}}_{Ł}(\Phi)$ are jointly satisfiable. Moreover, in the affirmative case, the values assigned to the linear variables in the first k rows of $\mathbf{x}$ encode a satisfying FP(Ł)-valuation for Φ.*

4. A Decision Procedure for FP(Ł)-Sat

To devise a decision procedure for FP(Ł)-Sat, we adopt a common strategy used in solving MILP problems with binary constraints. First, all the binary constraints are relaxed from "$b \in \{0,1\}$" to "$b \in [0,1]$". Then, integrality is iteratively reinstated for each binary variable, one at a time, within a solving routine, where each iteration splits the problem into two subproblems, leading to a branching computation.

The subproblems resulting from such a branching procedure are feasibility problems of continuous linear constraints, which are solved via a column generation strategy. In this way, the whole matrix equation in (8), which is exponential in the FP(Ł)-Sat input instance, does not need to be explicitly stored from the beginning of the procedure. In particular, columns associated with CPL-valuations (namely, those in A_{rt} and A_{rb}) are generated and stored only on demand during each linear feasibility solving process.

4.1. Relaxation and Reinstatement of Binary Constraints

Let L be the set of linear constraints derived from the linear formulation of FP(Ł)-Sat for an FP(Ł)-formula Φ. Set L includes the continuous constraints given by (8) along with a finite number of binary constraints, as established in Section 3. Let us denote the finite set of binary variables appearing in such formulation by B; and let $|B| = \beta \in \mathbb{N}$.

Then, $L^{\emptyset}$ is defined from L by relaxing its binary constraints, which may be achieved by extending (8) to include the constraints "$0 \leq b_\iota \leq 1$", for $\iota = 1, \ldots, \beta$. For each binary variable b_ι, it suffices to include the linear constraint "$b_\iota + y_{b_\iota} = 1$" in the standard formulation of (8), where y_{b_ι} is a fresh auxiliary linear variable subject to the nonnegativity constraint "$y_{b_\iota} \geq 0$". Then, A_{lt} is extended by adding one row for each new linear constraint and one column corresponding to its auxiliary variable. In total, A_{lt} will gain β new rows and β new columns. Matrix A_{lb} must also be extended to accommodate β additional columns and matrix A_{rt} must be extended to accommodate β additional rows; however, the definitions of their elements remain unchanged.

If the continuous linear constraints in $L^{\emptyset}$ are jointly infeasible, then Φ is FP(Ł)-unsatisfiable, as the infeasibility of a relaxed constraint "$b \in [0,1]$" implies, *a fortiori*, the infeasibility of the binary constraint "$b \in \{0,1\}$". Otherwise, as the feasibility of "$b \in [0,1]$" does not guarantee the feasibility of "$b \in \{0,1\}$", our proposed decision procedure continues to the next iteration where two new sets of constraints are defined. A binary variable $b_1 \in B$ is chosen and $L^{\emptyset}$ is expanded in two different ways by

$$L^{b_1=0} \doteq L^{\emptyset} \cup \{b_1 = 0\} \qquad \text{and} \qquad L^{b_1=1} \doteq L^{\emptyset} \cup \{b_1 = 1\},$$

branching the computation. These sets of constraints may also be put into standard form with simple adjustments to (8). Now, if both $L^{b_1=0}$ and $L^{b_1=1}$ are infeasible, Φ is FP(Ł)-unsatisfiable. Otherwise, nothing may be stated about the FP(Ł)-satisfiability status of Φ. In case both $L^{b_1=0}$ and $L^{b_1=1}$ are feasible, another binary variable $b_2 \in B \setminus \{b_1\}$ is chosen and two new sets of constraints are defined from each of $L^{b_1=0}$ and $L^{b_1=1}$, now branching the computation into the feasibility verification of four sets of constraints: $L^{b_1=0,b_2=0}$, $L^{b_1=0,b_2=1}$, $L^{b_1=1,b_2=0}$ and $L^{b_1=1,b_2=1}$. Ultimately, if only one of $L^{b_1=0}$ or $L^{b_1=1}$ is feasible, then only that set induces a new computation branch.

In general, for a set of constraints $L^{b_1=a_1,\ldots,b_l=a_l}$ (that could also be $L^{\emptyset}$), where $b_1, \ldots, b_l \in B$ and $a_1, \ldots, a_l \in \{0,1\}$, we may define

$$\begin{aligned} L^{b_1=a_1,\ldots,b_l=a_l,b_{l+1}=0} &\doteq L^{b_1=a_1,\ldots,b_l=a_l} \cup \{b_{l+1} = 0\} \text{ and} \\ L^{b_1=a_1,\ldots,b_l=a_l,b_{l+1}=1} &\doteq L^{b_1=a_1,\ldots,b_l=a_l} \cup \{b_{l+1} = 1\}, \end{aligned} \tag{9}$$

for some chosen $b_{l+1} \in B \setminus \{b_1, \ldots, b_l\}$. Formally, a *branch* is: a sequence of sets of constraints

$$\langle L^{\emptyset},\ L^{b_1=a_1},\ L^{b_1=a_1,b_2=a_2},\ \ldots,\ L^{b_1=a_1,\ldots,b_l=a_l} \rangle, \tag{10}$$

Algorithm 1 Branching Procedure for FP(Ł)-SAT

Require: An FP(Ł)-formula Φ
Ensure: SAT in case Φ is FP(Ł)-satisfiable, or UNSAT otherwise

Establish the set of relaxed constraints $L^{\emptyset}$ from Φ, where B is the set of original binary variables that were relaxed in $L^{\emptyset}$
$\mathfrak{B} \leftarrow B$
if $L^{\emptyset}$ is feasible **then**
    $\mathfrak{L} \leftarrow \{L^{\emptyset}\}$
else
    $\mathfrak{L} \leftarrow \emptyset$
end if
while $\mathfrak{B} \neq \emptyset$ **and** $\mathfrak{L} \neq \emptyset$ **do**
    Choose $b \in \mathfrak{B}$ ▷ nondeterministic step
    $\mathfrak{B} \leftarrow \mathfrak{B} \setminus \{b\}$
    for all $\mathcal{L} \in \mathfrak{L}$ **do**
        $\mathfrak{L} \leftarrow \mathfrak{L} \setminus \{\mathcal{L}\}$
        if $\mathcal{L}^{b=0}$ is feasible **then**
            $\mathfrak{L} \leftarrow \mathfrak{L} \cup \{\mathcal{L}^{b=0}\}$
        end if
        if $\mathcal{L}^{b=1}$ is feasible **then**
            $\mathfrak{L} \leftarrow \mathfrak{L} \cup \{\mathcal{L}^{b=1}\}$
        end if
    end for
end while
if $\mathfrak{L} = \emptyset$ **then**
    return UNSAT ▷ Φ is FP(Ł)-unsatisfiable
else
    return SAT ▷ Φ is FP(Ł)-satisfiable
end if

for some $l \in \{1, \ldots, \beta\}$ and $a_1, \ldots, a_l \in \{0, 1\}$, where $b_i \neq b_j$, for $i \neq j$; or the single-element sequence $\langle L^{\emptyset} \rangle$. Our procedure splits a branch only if its last element is a feasible set $L^{b_1=a_1,\ldots,b_l=a_l}$. When branching occurs, the original branch is replaced by two new branches that share the first $l+1$ elements with the original but have one new last element each, given by (9), namely $L^{b_1=a_1,\ldots,b_l=a_l,b_{l+1}=0}$ and $L^{b_1=a_1,\ldots,b_l=a_l,b_{l+1}=1}$. We will then only consider branches (10) in which:

- Either all sets of constraints are feasible, which we call an *open* branch;
- Or all but the last set of constraints are feasible, which we call a *closed* branch.

In each iteration, each current open branch is again split into two new branches. The branching procedure continues until no binary variable remains to be reinstated in any open branch, or until all branches are closed. In the former case, the assignment of linear variables that makes the last set of constraints feasible in any open branch also jointly satisfies the original set of constraints L along with the binary constraints "$b_1 \in \{0, 1\}$", …, "$b_\beta \in \{0, 1\}$". This assignment determines a satisfiable FP(Ł)-valuation v for Φ, given by $v(\Psi) = x_\Psi$ for all subformulas Ψ of Φ. In the latter case, Φ is FP(Ł)-unsatisfiable, since the infeasibility of the last set of constraints in all branches implies that no assignment satisfies L and the binary constraints together.

Algorithm 1 realizes such a procedure where, in each iteration, the same binary variable $b_{l+1} \in B \setminus \{b_1, \ldots, b_l\}$ is chosen to have integrality reinstated in all new sets of constraints derived from the splitting of open branches. So, we only need to keep track of a single sequence of reinstated variables, $b_1, b_2 \ldots$, during the execution of the procedure.

Given an FP(Ł)-formula Φ, Algorithm 1 first establishes the set $L^{\emptyset}$ of relaxed linear constraints (line 1) and initializes the algorithm variable $\mathfrak{B}$, which stores the set of binary variables that have not yet been reinstated in the procedure. Initially, $\mathfrak{B}$ is set to B (line 2). Then, the algorithm variable $\mathfrak{L}$, which stores the last element (set of constraints) in each of the currently open branches, is initialized to either $\{L^{\emptyset}\}$ or $\emptyset$, since, up to this point, $\langle L^{\emptyset} \rangle$ is the only branch determined in the procedure (lines 3–7). The main loop (lines 8–20) is executed as long as there are open branches and binary variables still to be reinstated.

In each iteration of the main loop, a binary variable $b \in \mathfrak{B}$ is nondeterministically chosen for reinstatement (line 9), and new branches are created by establishing $\mathcal{L}^{b=0}$ and $\mathcal{L}^{b=1}$ from each $\mathcal{L} \in \mathfrak{L}$, representing the last set of constraints in each currently open branch (lines 11–19). The sets $\mathfrak{B}$ and $\mathfrak{L}$ are updated by removing the reinstated variable b and the elements $\mathcal{L} \in \mathfrak{L}$, which will no longer be the last ones in an open branch (lines 10 and 12, respectively).

In the main loop, set $\mathfrak{L}$ is also updated by adding the newly established elements $\mathcal{L}^{b=0}$ and/or $\mathcal{L}^{b=1}$ if they are feasible sets (lines 13–18). In positive case, these sets of constraints become the last elements of the newly open branches. However, if they are infeasible, they become the last elements of newly closed branches, in which case they should not be included in $\mathfrak{L}$.

The algorithm terminates by: returning UnSat if no open branches remain, in which case $\mathfrak{L} = \emptyset$; or returning Sat if there are still open branches but no variables left to be reinstated, in which case $\mathfrak{L} \neq \emptyset$, but $\mathfrak{B} = \emptyset$ (lines 21–25).

Theorem 2. *Algorithm 1 terminates and returns a correct decision to FP(Ł)-Sat for any FP(Ł)-formula Φ given as input.*

Proof. Let L be the set of continuous constraints associated with Φ, as established in Section 3. Based on our discussion in this section, from $L^{\emptyset}$, Algorithm 1 develops branches by reinstating integrality to the relaxed variables in $L^{\emptyset}$ until each branch is closed ($\mathfrak{L} = \emptyset$) or can no longer be developed due to the lack of binary variables to be reinstated ($\mathfrak{L} \neq \emptyset$, but $\mathfrak{B} = \emptyset$).

Algorithm 1 is guaranteed to terminate because: the condition in the loop on lines 8–20 eventually becomes false, as $\mathfrak{B}$ is initialized as a finite set and decreases in one element in each iteration due to the instructions in lines 9 and 10; and the loop on lines 11–19 operates within an always finite set $\mathfrak{L}$, since $\mathfrak{L}$ is initialized as a singleton and at most doubles in size within this loop.

For the correctness, let $\mathcal{L}$ be an element in a branch containing the relaxed constraint "$b \in [0, 1]$", where b is a relaxed binary variable. Such a branch may be further split by Algorithm 1 into two new branches, $\mathcal{L}^{b=0}$ and $\mathcal{L}^{b=1}$. Note that if both $\mathcal{L}^{b=0}$ and $\mathcal{L}^{b=1}$ are infeasible, then the original set $\mathcal{L}$ and the binary constraint "$b \in \{0, 1\}$" are jointly infeasible. However, if one of the sets $\mathcal{L}^{b=0}$ and $\mathcal{L}^{b=1}$ is feasible (they cannot both be feasible), then $\mathcal{L}$ and constraint "$b \in \{0, 1\}$" are jointly feasible. Thus, any fully developed open branch ensures the joint feasibility of L along with the binary constraints determined from Φ (as established in Section 3). On the other hand, if all fully developed branches turn out to be closed, then L and the binary constraints determined from Φ cannot be jointly feasible. Correctness then follows from Theorem 1. □

4.2. Feasibility of Linear Constraints

The branching procedure described above depends on checking the feasibility of certain sets of linear constraints (see Algorithm 1, lines 13 and 16). Our goal now is to design a method for determining whether such sets of constraints are feasible. To this end, we adopt a strategy based on linear programming combined with column generation. Such a strategy has been successfully adopted previously to solve the classical probabilistic satisfiability problem (Psat) [11].

The sets of linear constraints we consider are derived from adjustments to (8), made to accommodate relaxed and/or reinstated binary variables, as discussed earlier. We fix the notation:

$$A_l = \begin{bmatrix} A_{lt} \\ A_{lb} \end{bmatrix} \quad \text{and} \quad A_r = \begin{bmatrix} A_{rt} \\ A_{rb} \end{bmatrix}.$$

A set of constraints for which we wish to decide feasibility—given by adjustments to (8)—may be written as:

$$[A_l\ A_r]\begin{bmatrix}\mathbf{x}\\ \mathbf{p}\end{bmatrix} = \begin{bmatrix}\mathbf{b}\\ \mathbf{0}_k\\ 1\end{bmatrix} \qquad \mathbf{x}, \mathbf{p} \geq 0 \tag{11}$$

The columns of A_l correspond to the linear variables in the encoding of the representation of Ł-operators through MILP constraints. Meanwhile, the columns of A_r correspond to evaluations of the CPL-formulas $\psi_1, \ldots, \psi_k$, occurring in the FP(Ł)-formula $\Phi[P\psi_1, \ldots, P\psi_k]$, through CPL-valuations.

Let us assume that matrix $[A_l\ A_r]$ has M rows and N columns. We define the following linear program, where I_M denotes the $M \times M$ identity matrix and $\mathbf{i}_M = [i_1\ \cdots\ i_M]^T$ is a column vector with M fresh linear variables:

$$\begin{array}{l}\text{minimize } i_1 + \cdots + i_M\\ \text{subject to}\end{array}$$

$$[I_M\ A_l\ A_r]\begin{bmatrix}\mathbf{i}_M\\ \mathbf{x}\\ \mathbf{p}\end{bmatrix} = \begin{bmatrix}\mathbf{b}\\ \mathbf{0}_k\\ 1\end{bmatrix} \qquad \mathbf{i}_M, \mathbf{x}, \mathbf{p} \geq 0 \tag{12}$$

We assume that the MILP constraints encoding the behavior of Ł-operators are formulated such that $\mathbf{b} \geq 0$, which may always be ensured. Under this assumption, the linear program (12) admits at least one basic feasible solution given by ${\mathbf{i}_\mathbf{M}}^T = [\mathbf{b}^T\ \mathbf{0}_k^T\ 1]$ and $\mathbf{x} = \mathbf{p} = \mathbf{0}_N$.

Lemma 1. *The linear program* (12) *has optimal value* 0 *if, and only if, the set of constraints* (11) *is feasible.*

Proof. Assume that the optimal value of linear program (12) is 0 for a feasible solution given by $\mathbf{i}_M$, $\mathbf{x}$ and $\mathbf{p}$. Then, we should have $\mathbf{i}_M = \mathbf{0}_M$ and the values of $\mathbf{x}$ and $\mathbf{p}$ make (11) feasible. Conversely, if (11) is feasible, then there exist nonnegative vectors $\mathbf{x}$ and $\mathbf{p}$ satisfying it. By taking $\mathbf{i}_M = \mathbf{0}$, we obtain a feasible solution to (12) with objective value 0. □

Linear program (12) may have an undesirable exponential size in the number of propositional variables $X_1, \ldots, X_n$ occurring in the CPL-formulas $\psi_1, \ldots, \psi_k$, since there are 2^n partial CPL-valuations over them. To circumvent this issue, one may instead search for the first instance in the following sequence of linear programs that achieves an optimal value of 0, indexed by $t = 1, \ldots, 2^n$:

$$\begin{array}{l}\text{minimize } i_1 + \cdots + i_M\\ \text{subject to}\end{array}$$

$$[I_M\ A_l\ A_r^{(t)}]\begin{bmatrix}\mathbf{i}_M\\ \mathbf{x}\\ \mathbf{p}^{(t)}\end{bmatrix} = \begin{bmatrix}\mathbf{b}\\ \mathbf{0}_k\\ 1\end{bmatrix} \qquad \mathbf{i}_M, \mathbf{x}, \mathbf{p}^{(t)} \geq 0 \tag{13}$$

In each linear program (13), for a fixed t, matrix $A_r^{(t)}$ contains only t selected columns from A_r, not necessarily in the same order. The corresponding variable vector $\mathbf{p}^{(t)}$ contains the t linear variables from $\mathbf{p}$ associated with those columns, arranged consistently with $A_r^{(t)}$. Also, a matrix $A_r^{(t+1)}$ contains exactly t columns that are identical to those in $A_r^{(t)}$, and therefore includes one new column.

By Carathéodory's Theorem (see e.g. [12]), if (12) has an optimal feasible solution, then it must have an optimal feasible solution with at most M nonzero values across $\mathbf{i}_M$, $\mathbf{x}$ and $\mathbf{p}$. Therefore, if (12) has

an optimal value of 0, then (13) might also have an optimal value of 0, possibly for some t much smaller than 2^n.

The simplex method for solving linear programs (see e.g. [13]) iteratively searches precisely for such solutions with a limited number of nonzero values. At each iteration, it selects a new column from the coefficient matrix, which is currently associated with a zero-valued linear variable, to *enter the basis*—that is, such linear variable is meant to possibly have nonzero value in the next iteration. This operation replaces a column that is currently in the basis with the new one, potentially improving the value of the objective function.

The decision to insert a column into the basis is guided by its *reduced cost*, which indicates whether the replacement operation may improve the objective function value in the current solution. In minimization problems, a negative reduced cost signals that introducing a column into the basis may lead to a lower objective function value. In the context of linear program (12), the reduced cost of a column from A_r is given by

$$\bar{c}_j = -\mathbf{c}_B^T B^{-1} (A_r)_j, \tag{14}$$

where $\bar{c}_j$ is the reduced cost of $(A_r)_j$, the jth column of A_r; B is the basis—the submatrix of the coefficient matrix whose columns correspond to the only M linear variables allowed to have nonzero values in the current solution—and $\mathbf{c}_B$ is the column vector of the objective function coefficients associated with the columns of B, in order.

Our proposal is to solve the sequence of linear programs (13), increasing t iteratively, until we either obtain an instance with optimal value 0 or exhaust all columns with negative reduced cost in A_r. For each linear program in the sequence, given by $A_r^{(t)}$, with optimal value greater than 0, we select a column from A_r with negative reduced cost and that does not appear in $A_r^{(t)}$. This new column is added to $A_r^{(t)}$, so establishing $A_r^{(t+1)}$.

In order to a column vector A with entries in $\{0, 1\}$ to be eligible to be added to $A_r^{(t)}$, it must simultaneously satisfy equation

$$\mathbf{c}_B^T B^{-1} A > 0 \tag{15}$$

and stand for a CPL-valuation for formulas $\psi_1, \ldots, \psi_k$. A possibility for searching for column A is to encode inequality (15) as a (pseudo-)Boolean satisfiability instance and use (pseudo-)Boolean satisfiability solving techniques to find a CPL-valuation for $\psi_1, \ldots, \psi_k$ that satisfies (15).

5. Implementation, Experiments, and Results

In order to derive statistical properties of FP(Ł)-SAT and of our algorithm, we have developed a publicly available[4] C++ implementation with the following specifications:

- The SoPlex callable library[5] is used to solve linear programs [14];
- Binary variables in $\mathfrak{B}$ are named according to the subformulas to which they are related, and the choice in line 9 of Algorithm 1 follows the increasing lexicographic order of the variable names;
- Column generation is performed by encoding equation (15) as a pseudo-Boolean instance and calling an external pseudo-Boolean solver; in our experiments, we use MiniSat+[6] [15].

Given the many possible formula structures in the FP(Ł) language, we focus on a simple fragment based on known properties of CPL and Ł systems. We start from basic modal formulas of the form $P\varphi$, where φ is in conjunctive normal form (CNF), and build modal FP(Ł)-formulas with the structure of *simple Ł-clausal form formulas* [16], which are given by

$$\bigwedge_{1 \leq i \leq \ell_2} \left(\bigoplus_{1 \leq j \leq \ell_1(i)} l_{ij} \right), \tag{16}$$

[4] https://github.com/spreto/fplsol
[5] https://soplex.zib.de
[6] http://minisat.se/MiniSat+.html

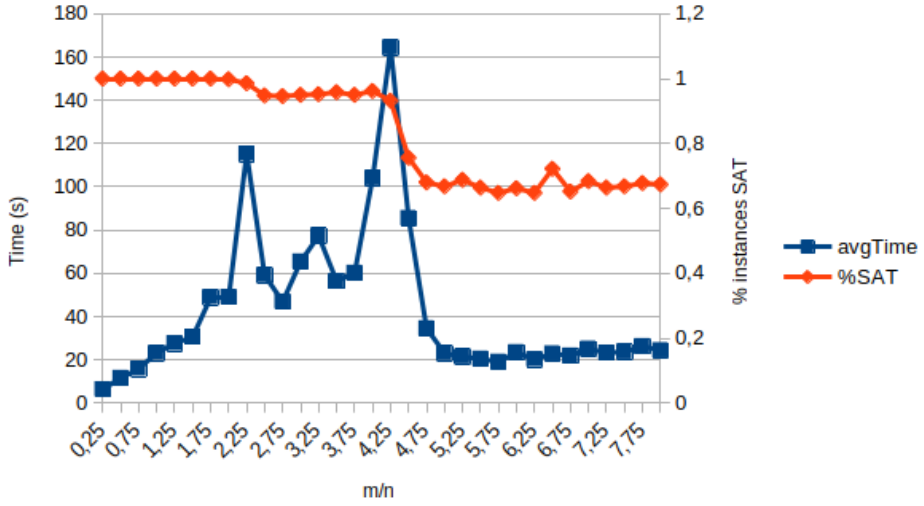

Figure 1: Average running time and percentage of satisfiable instances.

Table 3
Percentage of satisfiable instances for ratios $m/n \geq 4.25$

m/n	4.25	4.5	4.75	5	5.25	5.5	5.75	6	6.25	6.5	6.75	7	7.25	7.5	7.75	8
%SAT	0.932	0.756	0.68	0.668	0.688	0.664	0.648	0.662	0.648	0.722	0.652	0.684	0.664	0.668	0.678	0.674

where $i, j, \ell_2, \ell_1(i) \in \mathbb{N}$ and the l_{ij} are Ł-literals (positive or negated propositional variables in $\mathcal{L}_{Ł}$). Thus, instead of Ł-propositional variables in (16), we use basic modal formulas $P\varphi$, where φ is a CNF formula in $\mathcal{L}_{\mathrm{CPL}}$.

For the experimental evaluation, we randomly generate instances of the form (16), fixing $\ell_1(i) = \ell_2 = 3$ for all i. The basic modal formulas $P\varphi$ (used instead of Ł-propositional variables) have a 50% chance of being negated. Each CNF formula φ consists of m 3-clauses, where CPL-propositional variables are chosen uniformly from a fixed set of $n = 120$ variables and have a 50% chance of being negated.

We run experiments on 500 randomly generated instances for each configuration, with $m \in \{30, 60, \ldots, 960\}$, so that the ratio $\frac{m}{n}$ varies in steps of 0.25. Each run is subject to a timeout of two hours of user CPU time; instances exceeding this limit are reported separately.

While the satisfiability of CNF formulas is well known to be an **NP**-complete problem [17], the satisfiability of simple Ł-clausal form formulas is solvable in linear time [16, Lemma 2]. Even under this restriction on the class of FP(Ł)-formulas, we observe a clear phase transition and a well-defined behavior of the average running time. This indicates that these phenomena already emerge in a simple and controlled setting. The experimental framework introduced here therefore serves as a baseline for future studies exploring richer structures and additional parameters.

Figure 1 reports the average running time of our solver (excluding timeout instances), together with the percentage of satisfiable formulas (also excluding timeout instances), as a function of the ratio $\frac{m}{n}$. Timeout occurrences were rare in our experiments—only 21 out of a total of 16,000 instances—so their exclusion does not impact the statistical validity of the results.

As the ratio $\frac{m}{n}$ increases, the plot exhibits a well-defined phase transition behaviour. Throughout the regime where almost all generated instances are satisfiable, the average running time follows an approximately linear trend. Within this predominantly satisfiable regime, however, two values of $\frac{m}{n}$ stand out by exhibiting abrupt increases in the average running time.

The first such increase occurs at $\frac{m}{n} = 2.25$, where the percentage of satisfiable instances slightly drops for the first time, from nearly 100% to approximately 95%. After this point, the average running time returns to its previous linear trend. As the ratio further increases, a second abrupt peak in the

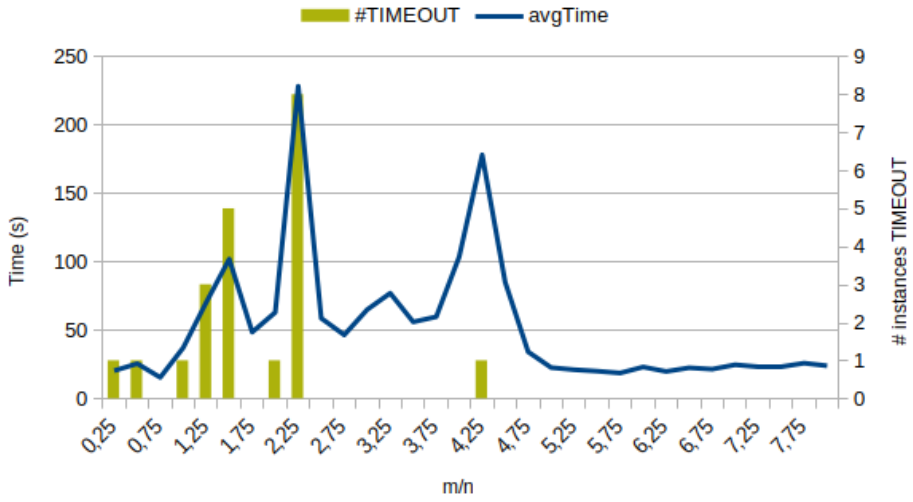

Figure 2: Timeout distribution and average running time considering timeouts.

running time is observed at $\frac{m}{n} = 4.25$, which corresponds to the last point at which the percentage of satisfiable instances remains close to 95%. This is followed by a more pronounced transition, in which the percentage of satisfiable instances decreases sharply to around 67% (see Table 3), while the average running time drops abruptly.

We can give an explanation to as why, for $\frac{m}{n} \geq 4.25$, the percentage of satisfiable instances decreases sharply to around 67%, and not to around 0% as in the classical 3-SAT problem, by the following argument: remember that the basic modal formulas $P\varphi$ have 50% chances of being negated. Thus, when φ becomes unsatisfiable, $P\varphi$ will have value 0 and then $\neg P\varphi$ will take value 1, whence an FP(Ł)-formula of the form $\bigoplus_{1\leq j\leq 3}(\neg)P\varphi_j$, will be satisfiable whenever there is at least one φ_j unsatisfiable and $P\varphi_j$ occurs negated. Since our formulas are Ł-disjunctions of 3 literals, the probability that at least one basic modal formula $P\varphi$ appears negated has probability $1-(0.5)^3 = 0.875$. In addition, these formulas are taken conjunctively (minimum operator), because the formulas we are checking are of the form

$$\bigwedge_{1\leq i\leq 3}\left(\bigoplus_{1\leq j\leq 3}(\neg)P\varphi_{i,j}\right),$$

and therefore the probability of formulas like the above that are satisfiable because of the unsatisfiability of some $\varphi_{i,j}$ is $(0.875)^3 \approx 0.669$.

The presence of two distinct peaks in the average running time suggests different sources of computational hardness, with difficult instances already emerging in a pre-critical point, before the main decrease in satisfiability.

Figures 2 and 3 further clarify the source of computational hardness. All timeout occurrences are confined to the predominantly satisfiable regime, up to $\frac{m}{n} = 4.25$, with no timeouts observed during or after the main decrease in satisfiability. This indicates that the hardest instances arise before the transition, rather than in the regime where unsatisfiable formulas become more frequent.

A more detailed inspection separating satisfiable and unsatisfiable instances further shows that the former are generally harder than the latter. As shown in Figure 3, the average running time for satisfiable instances consistently dominates that of unsatisfiable ones across all values of $\frac{m}{n}$. Although the running time for satisfiable instances decreases after the phase transition, it remains higher than that for unsatisfiable formulas. Unsatisfiable instances exhibit their highest average running time at the point where they first appear ($\frac{m}{n} = 2$), after which their resolution becomes progressively easier and stabilizes.

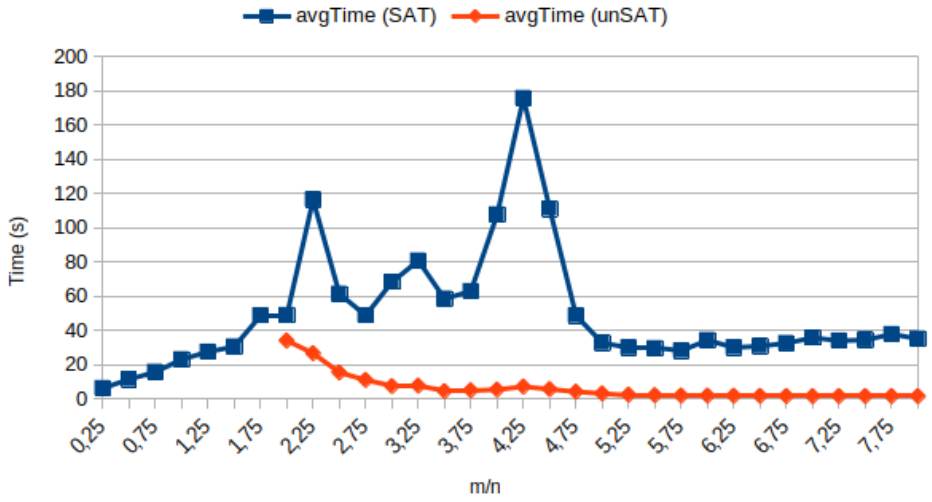

Figure 3: Average running times for satisfiable and unsatisfiable instances.

6. Conclusions and Future Work

In this work, we presented a detailed algorithm for solving the FP(Ł)-SAT problem via a translation of instances into linear constraints. Moreover, we introduced a parameterized class of FP(Ł)-SAT instances for which we were able to observe a phase transition phenomenon and distinctive running-time patterns.

As future work, further empirical experiments may be conducted by exploring different parameter configurations and variations, as well as other formula structures. Other heuristics for the iterative choice of binary variables in $\mathfrak{B}$ (Algorithm 1, line 9), beyond the lexicographic order, may also be investigated. In addition, algorithms for deciding validity and entailment in FP(Ł) deserve further investigation, noting that in Łukasiewicz logic the entailment problem is not directly related to satisfiability as in classical logic.

Łukasiewicz logic has been used to model neural network computations and to perform formal verification of their properties [18, 19, 20]. Since FP(Ł) provides a natural framework for expressing fuzzy notions of probability, the tools developed in this work may, from a practical perspective, play a role in the interpretation and formal verification of probabilistic neural models.

Acknowledgments

This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the University of São Paulo and the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2019/07665-4. This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2024/19144-7. Authors are partially supported by the MSCA project SemPER - SEMantics, Proofs and Effective Reasoning (grant agreement number 101299559). Finger was partially supported by CNPQ Grant PQ 302963/2022-7. Flaminio acknowledges partial support by the Spanish project SHORE (PID2022-141529NB-C21) while Godo, Manyà and Preto are partially supported by the Spanish project LINEXSYS (PID2022-139835NB-C21), both funded by MCIU/AEI/10.13039/501100011033. Preto acknowledges that this study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2026/02601-1.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

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

[2] P. Hájek, Metamathematics of Fuzzy Logic, Trends in Logic, Springer, Dordrecht, 1998.

[3] P. Hájek, L. Godo, F. Esteva, Fuzzy logic and probability, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 1995, pp. 237–244.

[4] P. Baldi, P. Cintula, C. Noguera, Classical and fuzzy two-layered modal logics for uncertainty: Translations and proof-theory, International Journal of Computational Intelligence Systems 13 (2020) 988–1001.

[5] R. Cignoli, I. D'Ottaviano, D. Mundici, Algebraic Foundations of Many-Valued Reasoning, Trends in Logic, Springer Netherlands, 2000.

[6] T. Flaminio, S. Preto, S. Ugolini, Reasoning about probability via continuous functions., in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, 2023, pp. 282–290.

[7] P. Hájek, S. Tulipani, Complexity of fuzzy probability logics, Fundamenta Informaticae 45 (2001) 207–213. doi:10.3233/FUN-2001-45304.

[8] G. Georgakopoulos, D. Kavvadias, C. H. Papadimitriou, Probabilistic satisfiability, Journal of Complexity 4 (1988) 1–11. doi:10.1016/0885-064X(88)90006-4.

[9] M. Finger, D. M. Gabbay, Adding a temporal dimension to a logic system, Journal of Logic, Language and Information 1 (1992) 203–233. doi:10.1007/bf00156915.

[10] R. Hähnle, Many-valued logic and mixed integer programming, Annals of Mathematics and Artificial Intelligence 12 (1994) 231–263.

[11] M. Finger, G. De Bona, Probabilistic satisfiability: algorithms with the presence and absence of a phase transition, Annals of Mathematics and Artificial Intelligence 75 (2015) 351–379. URL: http://dx.doi.org/10.1007/s10472-015-9466-6. doi:10.1007/s10472-015-9466-6.

[12] A. Brøndsted, An Introduction to Convex Polytopes, Springer-Verlag, Berlin, 1983.

[13] D. Bertsimas, J. N. Tsitsiklis, Introduction to linear optimization, Athena Scientific, Nashua, NH, 1997.

[14] S. Bolusani, M. Besançon, K. Bestuzheva, A. Chmiela, J. Dionísio, T. Donkiewicz, J. van Doornmalen, L. Eifler, M. Ghannam, A. Gleixner, C. Graczyk, K. Halbig, I. Hedtke, A. Hoen, C. Hojny, R. van der Hulst, D. Kamp, T. Koch, K. Kofler, J. Lentz, J. Manns, G. Mexi, E. Mühmer, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, M. Turner, S. Vigerske, D. Weninger, L. Xu, The SCIP Optimization Suite 9.0, Technical Report, Optimization Online, 2024. URL: https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/.

[15] N. Eén, N. Sörensson, Translating pseudo-boolean constraints into SAT, Journal on Satisfiability, Boolean Modelling and Computation 2 (2006) 1–26.

[16] M. Bofill, F. Manyà, A. Vidal, M. Villaret, New complexity results for Łukasiewicz logic, Soft Computing 23 (2019) 2187–2197. doi:10.1007/s00500-018-3365-9.

[17] R. M. Karp, Reducibility among Combinatorial Problems, Springer US, Boston, MA, 1972, pp. 85–103.

[18] S. Preto, M. Finger, Effective reasoning over neural networks using Łukasiewicz logic, in: P. Hitzler, M. Kamruzzaman Sarker, A. Eberhart (Eds.), Compendium of Neurosymbolic Artificial Intelligence, volume 369 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 609–630. doi:10.3233/FAIA230160.

[19] S. Preto, M. Finger, Proving properties of binary classification neural networks via Łukasiewicz logic, Logic Journal of the IGPL 31 (2023) 805–821. doi:10.1093/jigpal/jzac050.
[20] S. Preto, F. Manyà, M. Finger, Benchmarking Łukasiewicz logic solvers with properties of neural networks, in: 2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL), IEEE, 2023, pp. 158–163.