Soft Computing (2019) 23:2187–2197
https://doi.org/10.1007/s00500-018-3365-9

FOCUS

New complexity results for Łukasiewicz logic

Miquel Bofill[1] · **Felip Manyà**[2] · **Amanda Vidal**[3] · **Mateu Villaret**[1]

Published online: 11 July 2018

Abstract
One aspect that has been poorly studied in multiple-valued logics, and in particular in Łukasiewicz logic, is the generation of instances of varying difficulty for evaluating, comparing and improving satisfiability solvers. With the ultimate goal of finding challenging benchmarks for Łukasiewicz satisfiability solvers, we start by defining a natural and intuitive class of clausal forms (simple Ł-clausal forms) and studying their complexity. Since we prove that the satisfiability problem of simple Ł-clausal forms can be solved in linear time, we then define two new classes of clausal forms (Ł-clausal forms and restricted Ł-clausal forms) that truly exploit the non-lattice operations of Łukasiewicz logic and whose satisfiability problems are NP-complete when clauses have at least three literals, and admit linear-time algorithms when clauses have at most two literals. We also define an efficient satisfiability preserving translation of Łukasiewicz logic formulas into Ł-clausal forms. Finally, we describe a random generator of Ł-clausal forms and report on an empirical investigation in which we identify an easy-hard-easy pattern and a phase transition phenomenon for Ł-clausal forms.

Keywords Łukasiewicz logics · Clausal forms · Complexity · Instance generator

1 Introduction

The proof theory of multiple-valued logics, as well as its complexity, has been deeply studied for a wide variety of logics (Aguzzoli et al. 2005; Hájek 1998; Metcalfe et al. 2009, 2005). However, the development of satisfiability solvers has received less attention despite the fact that, without competitive solvers, it is extremely difficult to apply multiple-valued logics to solve real-world problems.

In the quest for developing fast satisfiability solvers, it is crucial to have both random and structured challenging instances for evaluating and comparing solvers, as happens in close areas such as Boolean satisfiability testing and constraint programming.

Given the recent development of satisfiability modulo theory-based (SMT based) SAT solvers for multiple-valued logics (Ansótegui et al. 2012, 2015, 2016; Vidal 2016; Vidal et al. 2012), as well as the need of empirically evaluating and comparing them with other existing approaches, the objective of this paper is to develop suitable benchmarks for satisfiability testing in Łukasiewicz logic, as well as analyze their complexity from both a theoretical and practical perspective.

More specifically, we begin this paper by analyzing how the conjunctive normal forms (CNFs) used by SAT solvers can be extended to Łukasiewicz logics by really making use of the non-lattice operations (while still being restricted to the logical language).[1] In a first very natural attempt, we replace the classical disjunction in Boolean CNFs with Łukasiewicz strong disjunction and interpret negation using Łukasiewicz negation. Interestingly enough, it turns out that the satisfiability problem of these clausal forms can be solved in linear time in the length of the formula, regardless of the size of the clauses and the cardinality of the truth value set (assuming it

Communicated by C. Noguera.

✉ Felip Manyà
felip@iiia.csic.es

Miquel Bofill
miquel.bofill@udg.edu

[1] Dept. IMAE, Universitat de Girona, Carrer de la UdG 6, 17003 Girona, Spain

[2] Artificial Intelligence Research Institut (IIIA, CSIC) Campus UAB, Carrer de Can Planas, Zona 2, 08193 Bellaterra, Spain

[3] Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic

[1] For what concerns CNFs as conjunctions of disjunctions of literals see, e.g., Mundici and Olivetti (1998). On the other hand, resolution for a particular class of formulas defined by means of integer constants has also been studied in Wagner (1998).

is greater than two). Recall that, in contrast, deciding the satisfiability of Boolean CNFs is NP-complete when there are clauses with at least three literals (Garey and Johnson 1979). In multiple-valued logics, it can be even NP-complete when there are clauses with at least two literals (Beckert et al. 1999, 2000; Manyà 2000).

The focus is then moved to producing instances that are on the one hand computationally challenging, and so, rich enough to model complex notions, and on the other, generated in a simple and regular way that opens the door to a systematic study (complexity, resolution methods, etc). With this aim, we introduce a new class of clausal forms, called Łukasiewicz (Ł-)clausal forms, that are CNFs in which, besides replacing classical disjunction with Łukasiewicz strong disjunction, we allow negations above the literal level, i.e., clauses are strong disjunctions of terms, and terms are either literals or negated strong disjunctions of literals. We show that, in this case, 3-SAT is NP-complete, while 2-SAT has linear-time complexity.[2] Hence, these problems have the same complexity as their Boolean counterparts. We also propose a Tseitin-like transformation (Tseitin 1968) that translates any Łukasiewicz logic formula into a satisfiability preserving Ł-clausal form, showing that Ł-clausal forms are in fact a normal form for Łukasiewicz logic. Moreover, we introduce a fragment of Ł-clausal forms, called restricted Ł-clausal forms, and show that their 3-SAT problem is NP-complete and their 2-SAT problem admits linear-time algorithms.

Moreover, we report on an empirical investigation in which we identify—when testing the satisfiability of randomly generated Ł-clausal forms having a fixed number of literals per clause with SMT and mixed integer programming (MIP) solvers—an easy-hard-easy pattern and a phase transition phenomenon as the clause-to-variable ratio varies. It turns out that there is a point where the hardest instances occur. Such a point is between an under-constrained region where the instances are almost surely satisfiable, and an over-constrained region where the instances are almost surely unsatisfiable. In the transition region, there is a threshold where roughly half of the instances are satisfiable, and half of the instances are unsatisfiable. So, we have developed a generator that is able to produce both satisfiable and unsatisfiable instances of varying difficulty.

This paper is an extended and improved version of Bofill et al. (2015b). The new contributions are the introduction of restricted Ł-clausal forms and the complexity analysis of satisfiability problems, the satisfiability preserving translation of Łukasiewicz logic formulas into Ł-clausal forms, and a more extensive experimental investigation that considers both SMT and MIP solvers.

The paper is structured as follows. Section 2 defines basic concepts in Łukasiewicz logics. Section 3 defines three types of clausal forms with Łukasiewicz strong disjunction and Łukasiewicz negation. We show that the satisfiability problem is decidable in linear time for the first type of clausal forms but is NP-complete for the other types. Section 4 defines an efficient satisfiability preserving translation of Łukasiewicz logic formulas into Ł-clausal forms. Section 5 describes the random generator of Ł-clausal forms and reports on the conducted empirical investigation. Section 6 concludes and points out future research directions.

2 Preliminaries

In this section, we introduce the theoretical basis necessary for the development of the rest of the paper. For a deeper insight into many-valued logics, see, e.g., Cintula et al. (2011).

Definition 1 A ***propositional language*** is a pair $\mathbb{L} = \langle \Theta, \alpha \rangle$, where Θ is a set of logical connectives and $\alpha : \Theta \to \mathbb{N}$ defines the arity of each connective. Connectives with arity 0 are called constants. A language $\langle \Theta, \alpha \rangle$ with a finite set of connectives $\Theta = \{\theta_1, \ldots, \theta_r\}$ is denoted by $\langle \theta_1/\alpha(\theta_1), \ldots, \theta_r/\alpha(\theta_r) \rangle$.

Given a set of propositional variables $\mathcal{V}$, the set $L_\mathcal{V}$ of $\mathbb{L}$-formulas over $\mathcal{V}$ is inductively defined as the smallest set with the following properties: (i) $\mathcal{V} \subseteq L_\mathcal{V}$; (ii) if $\theta \in \Theta$ and $\alpha(\theta) = 0$, then $\theta \in L_\mathcal{V}$; and (iii) if $\phi_1, \ldots, \phi_m \in L_\mathcal{V}, \theta \in \Theta$ and $\alpha(\theta) = m$, then $\theta(\phi_1, \ldots, \phi_m) \in L_\mathcal{V}$.

Definition 2 A ***many-valued logic*** $\mathcal{L}$ is a tuple $\langle \mathbb{L}, N, A, D \rangle$ where $\mathbb{L} = \langle \Theta, \alpha \rangle$ is a propositional language, N is a truth value set, A is an interpretation of the operation symbols that assigns to each $\theta \in \Theta$ a function $A_\theta : N^{\alpha(\theta)} \to N$ and $D \subset N$ is a set of designated truth values.

The set of designated truth values from the previous definition can be understood as those which affirm satisfaction.

Definition 3 Let $\mathcal{L}$ be a many-valued logic. An ***interpretation on*** $\mathcal{L}$ is a function $I : \mathcal{V} \to N$. I is extended to arbitrary formulas ϕ in the usual way:

1. If ϕ is a logical constant, then $I(\phi) = A_\phi$.
2. If $\phi = \theta(\phi_1, \ldots, \phi_r)$, then $I(\theta(\phi_1, \ldots, \phi_r)) = A_\theta(I(\phi_1), \ldots, I(\phi_r))$.

A formula ϕ is ***satisfiable*** iff there is an interpretation I such that $I(\phi) \in D$.

[2] When the number of literals per clause is fixed to k, the corresponding SAT problem is called k-SAT. In the following, when we say linear-time complexity we mean that the complexity is linear in the size of the formula.

Throughout this work, we focus on a particular family of many-valued logics: the finite-valued and infinitely-valued Łukasiewicz logics. These were born from the generalization of a three-valued logic proposed by J. Łukasiewicz in the early twentieth century and have been deeply studied both from theoretical and practical points of view. For a deeper study on these matters, see for instance (Cignoli et al. 2000; Hájek 1998).

The language of Łukasiewicz logic is given by

$$\mathbb{L}_{Łuk} = \langle 0/0, 1/0, \neg/1, \rightarrow/2, \wedge/2, \vee/2, \odot/2, \oplus/2\rangle.$$

We refer to $\neg$ as negation, $\rightarrow$ as implication, $\wedge$ as weak conjunction, $\vee$ as weak disjunction, $\odot$ as (strong) conjunction, and $\oplus$ as (strong) disjunction. We will use the standard notation to shorten products and powers, namely for $k \in \mathbb{N}$,

$$k\varphi := \underbrace{\varphi \oplus \cdots \oplus \varphi}_{k \text{ times}} \quad \text{and} \quad \varphi^k := \underbrace{\varphi \odot \ldots \odot \varphi}_{k \text{ times}}$$

where $0\varphi := 0$ and $\varphi^0 := 1$.

Definition 4 The ***infinitely-valued Łukasiewicz logic***, denoted by $\mathbf{[0, 1]_Ł}$, is the many-valued logic $\langle \mathbb{L}_{Łuk}, [0, 1], A_Ł, \{1\}\rangle$ where the interpretation of the operation symbols $A_Ł$ is given by:

$$\begin{aligned} A_\neg(x) &= 1 - x \\ A_\rightarrow(x, y) &= \min\{1, 1 - x + y\} \\ A_\wedge(x, y) &= \min\{x, y\} \\ A_\vee(x, y) &= \max\{x, y\} \\ A_\odot(x, y) &= \max\{0, x + y - 1\} \\ A_\oplus(x, y) &= \min\{1, x + y\} \end{aligned}$$

The ***n-valued Łukasiewicz logic***, denoted by $\mathbf{Ł_n}$, is the logic defined from the infinitely-valued Łukasiewicz logic by restricting the universe of evaluation to the set $N_n = \{0, \frac{1}{n-1}, \ldots, \frac{n-1}{n-1}\}$. That is to say, $\mathbf{Ł_n} = \langle \mathbb{L}_{Łuk}, N_n, A_Ł, \{1\}\rangle$. Note that the operations are well defined because N_n is a subalgebra of $[0, 1]$ with the interpretation of the operation symbols $A_Ł$ (for any operation A_* and any value/pair of values of N_n, the result of the A_* over this/these values also belongs to N_n).

The function interpreting negation is called Łukasiewicz negation, the function interpreting strong conjunction is called Łukasiewicz t-norm, the function interpreting implication is called its residuum, and the function interpreting strong disjunction is called Łukasiewicz t-conorm.

We say that a logic $\mathcal{L}$ is a Łukasiewicz logic if it is either $\mathbf{[0, 1]_Ł}$ or $\mathbf{Ł_n}$ for some natural number n.

Given a Łukasiewicz logic $\mathcal{L}$, we denote by $\mathrm{SAT}^{\mathcal{L}}$ the set of satisfiable formulas in $\mathcal{L}$; i.e.,

$$\mathrm{SAT}^{\mathcal{L}} = \{\varphi : I(\varphi) = 1 \text{ for some interpretation } I \text{ on } \mathcal{L}\}.$$

The problem of deciding whether or not a formula belongs to the set $\mathrm{SAT}^{\mathcal{L}}$ is called the **$\mathcal{L}$-satisfiability problem**.

We say that two formulas φ, ψ are ***equivalent*** (and write $\varphi \equiv \psi$) if, for each interpretation I, $I(\varphi) = I(\psi)$.

It is remarkable that, in Łukasiewicz logic, many operations enjoy interdefinability properties among them. In particular, Łukasiewicz logic can be equivalently formulated using only the constants, $\neg$ and $\rightarrow$. Let us include here two important relations between operations that we use throughout the paper:

$$x \rightarrow y \equiv \neg x \oplus y \qquad \neg(x \oplus y) \equiv \neg x \odot \neg y,$$

where the latter one is called De Morgan's law for the strong conjunction and strong disjunction.

It is worth mentioning that one of the reasons we focus on Łukasiewicz logics is because $\mathrm{SAT}^{\mathrm{Bool}} \subset \mathrm{SAT}^{\mathcal{L}}$. This is not the case for other relevant logics such as Gödel (G) and Product (Π), where $\mathrm{SAT}^{G} = \mathrm{SAT}^{\Pi} = \mathrm{SAT}^{\mathrm{Bool}}$ (Hájek 1998). So, while Boolean solvers suffice for deciding the satisfiability of propositional formulas of G and Π, specific solvers are needed to decide the satisfiability of propositional formulas of $\mathcal{L}$.

3 Łukasiewicz clausal forms

In Boolean satisfiability, benchmarks are commonly represented in conjunctive normal form (CNF), i.e., as a conjunction of clauses, where each clause is a disjunction of literals. This formalism is very convenient because state-of-the-art Boolean SAT solvers implement variants of the Davis–Putnam–Logemann–Loveland (DPLL) procedure (Davis et al. 1962), and DPLL requires the input in CNF. Hence, it seems reasonable to ask how Boolean CNFs could be adapted to Łukasiewicz logic in order to define challenging benchmarks for evaluating and comparing Łukasiewicz SAT solvers, as well as in order to develop DPLL-like procedures for Łukasiewicz logic.

3.1 Simple Ł-clausal forms

A first natural attempt to generalize Boolean CNF formulas is to replace classical disjunction with Łukasiewicz strong disjunction, and negation with Łukasiewicz negation:

$$\bigwedge_{1 \le i \le n} \left(\bigoplus_{1 \le j \le k_i} l_{ij} \right)$$

for $i, j, k_i, n \in \mathbb{N}$ and l_{ij} literals. Note that a clausal form can still be interpreted as a set (of clauses) due to the use of weak conjunction. In fact, every clause represents a constraint

and a SAT solver determines whether all the constraints are satisfied. Relevant questions like MaxSAT (Li and Manyà 2009), that counts the number of unsatisfied clauses, have this approach. The use of strong conjunction would serve more involved notions by combining degrees of satisfaction, instead of simply representing the total or partial satisfaction of a set of clauses.

In the following, we refer to these formulas as **simple Łukasiewicz clausal forms (simple Ł-clausal forms)** and denote by $mc(\varphi)$ the length of the shortest clauses in a simple Ł-clausal form φ. That is to say, if $\varphi = \bigwedge_{1\leq i\leq n}(\bigoplus_{1\leq j\leq k_i} l_{ij})$, then $mc(\varphi) = \min\{k_i : 1 \leq i \leq n\}$. Throughout this work, we assume that simple Ł-clausal forms are interpreted using a truth value set with at least three elements.

Unfortunately, the expressive power of these clausal forms is quite limited. As Lemma 2 shows, the satisfiability problem for simple Ł-clausal forms has linear-time complexity, contrarily to what happens in Boolean SAT, which is NP-complete when there are clauses with at least three literals. Hence, complex problems cannot be encoded using this formalism.

It is quite easy to prove (see Lemma 1) that any simple Ł-clausal form φ is always satisfiable if $mc(\varphi)$ is greater than two, or if $mc(\varphi) = 2$ and the cardinality of the truth value set is odd or infinite. Interestingly, there is a particular case, for finitely-valued logics, that is more subtle: when $mc(\varphi) = 2$ and the cardinality of the truth value set is even. In this case, deciding the satisfiability of a simple Ł-clausal form φ turns out to be equivalent to deciding the satisfiability, under Boolean semantics, of the subformula of φ containing exclusively the clauses of φ with length 2; i.e., it is equivalent to deciding the satisfiability of a Boolean 2-SAT instance. We denote such a subformula by $B_2(\varphi)$; i.e., if $\varphi = \bigwedge_{1\leq i\leq n} C_i = \bigwedge_{1\leq i\leq n}(\bigoplus_{1\leq j\leq k_i} l_{ij})$, then

$$B_2(\varphi) = \bigwedge_{C_i\in\varphi, k_i=2} (l_{i1} \vee l_{i2})$$

Example 1 The simple Ł-clausal form $\varphi = (x_1 \oplus x_2) \wedge (\neg x_1 \oplus x_2) \wedge (\neg x_1 \oplus \neg x_2) \wedge (x_1 \oplus x_3) \wedge (x_2 \oplus x_3) \wedge (x_1 \oplus \neg x_2 \oplus \neg x_3)$ is satisfiable in Ł$_4$ because the Boolean 2-SAT instance $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)$ is satisfiable. However, φ is unsatisfiable under Boolean semantics. Recall that $\text{SAT}^{\text{Bool}} \subset \text{SAT}^{\mathcal{L}}$.

The treatment of formulas with unit clauses (i.e., containing exactly one literal) is done by applying unit propagation (UP). UP consists in applying the unit literal rule until the empty clause is derived or a saturation state is reached. Applying the unit literal rule to an Ł-clausal form φ containing the unit clause l_i amounts to removing from φ all the clauses containing l_i, and removing in φ all the occurrences of $\neg l_i$. If we remove a literal from a unit clause, we obtain the bottom element $\bot$. We denote by $\text{UP}(\varphi)$ the formula obtained after applying UP to φ. $\text{UP}(\varphi)$ is either empty (meaning that it is satisfiable), a formula containing the bottom element $\bot$ (meaning that there exists a contradiction at the unitary level, and so, the formula is directly unsatisfiable) or $mc(\text{UP}(\varphi)) > 1$, and so we fall in one of the scenarios discussed above.

Formally, we can express the previous results as follows.

Lemma 1 *Let φ be a simple Ł-clausal form, and let $\mathcal{L}$ be a Łukasiewicz logic with a truth value set N such that $|N| > 2$. Then,*

1. *If $mc(\varphi) > 1$, φ belongs to $SAT^{\mathcal{L}}$ if one of the following conditions hold:*
   - *$|N| = 2s + 1$ for $s \geq 1$ or $|N| \geq \aleph_0$*
   - *$mc(\varphi) \geq 3$*
2. *If $mc(\varphi) = 2$ and $|N| = 2s+2$ for $s \geq 1$, then φ belongs to $SAT^{\mathcal{L}}$ iff $B_2(\varphi)$ belongs to SAT^{Bool}.*
3. *If $mc(\varphi) = 1$, then φ belongs to $SAT^{\mathcal{L}}$ iff $\text{UP}(\varphi)$ belongs to $SAT^{\mathcal{L}}$.*

Proof 1. It is easy to see that whenever $\frac{1}{2}$ belongs to the universe of evaluation (i.e., whenever $|N| = 2s + 1$ or $\mathcal{L}$ is the infinitely-valued Łukasiewicz logic), the interpretation that assigns $\frac{1}{2}$ to each variable satisfies any possible simple Ł-clausal form φ such that $mc(\varphi) \geq 2$. On the other hand, if $mc(\varphi) \geq 3$ and $|N| = 2s + 2$ for some $s \geq 1$ ($N = \{0, \frac{1}{2s+1}, \ldots, \frac{2s+1}{2s+1}\}$), the interpretation that assigns $\frac{s+1}{2s+1}$ to each variable x satisfies any clause of φ in $\mathcal{L}$. It is clear that $s + 1 < 2s + 1$, so $\frac{s+1}{2s+1} \in N$. On the other hand, by the definition of Łukasiewicz negation, the interpretation of $\neg x$ is $1 - \frac{s+1}{2s+1} = \frac{s}{2s+1}$, and it is routine to check that $\frac{s+1}{2s+1} > \frac{s}{2s+1} \geq \frac{1}{3}$. Thus, for any clause $l_1 \oplus \cdots \oplus l_k$ in φ, the interpretation of each one of its literals is greater than or equal to $\frac{1}{3}$. Since $mc(\varphi) \geq 3$, we have that the interpretation of each clause is always 1.

2. First suppose there is an interpretation I on $\mathcal{L}$ that satisfies φ, and recall that $\frac{1}{2} \notin N$. We can then define a Boolean interpretation I' that satisfies $B_2(\varphi)$ by letting

$$I'(x) = \begin{cases} 1 & \text{if } I(x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

For each binary clause $l_1 \oplus l_2$ from φ, at least one of the strict inequalities $I(l_1) > \frac{1}{2}$ or $I(l_2) > \frac{1}{2}$ must hold in order that I satisfies the clause. So, we can assume, without loss of generality, that $I(l_1) > \frac{1}{2}$. If l_1 is a positive literal (equal to a variable x_1), then by definition $I'(x_1) = 1$, and so, $I(x_1 \vee l_2) = 1$. Otherwise ($l_1 = \neg x_1$), it holds

that $I(x_1) < \frac{1}{2}$. Then, by definition, $I'(x_1) = 0$. So $I'(\neg x_1) = 1$, and thus $I'(\neg x_1 \vee l_2) = 1$.
To prove the other direction, let $|N| = 2s + 2$ for some $s \geq 1$, and suppose I is an interpretation on $\{0, 1\}$ that satisfies $B_2(\varphi)$. Then, let I' be the interpretation in $\mathcal{L}$ defined by

$$I'(x) = \begin{cases} \frac{s+1}{2s+1} & \text{if } I(x) = 1 \\ 1 - \frac{s+1}{2s+1} & \text{otherwise} \end{cases}$$

As was proven in 1, this interpretation satisfies all the clauses of length at least 3, so we just need to check that the binary clauses from φ are also satisfied. Let $l_1 \oplus l_2$ be a binary clause of φ. Then, $l_1 \vee l_2$ is a clause from $B_2(\varphi)$, and thus, $I(l_1 \vee l_2) = 1$. Without loss of generality, assume that $I(l_1) = 1$. If it is a positive literal ($l_1 = x_1$), then $I'(x_1) = \frac{s+1}{2s+1}$ and $I'(l_2) \geq 1 - \frac{s+1}{2s+1}$, so $I'(x_1 \oplus l_2) \geq \frac{s+1}{2s+1} + 1 - \frac{s+1}{2s+1} = 1$. Otherwise, $I(\neg x_1) = 1$ implies that $I(x_1) = 0$, and thus, $I'(x_1) = 1 - \frac{s+1}{2s+1}$. Then again, $I'(\neg x_1 \oplus l_2) \geq 1 - (1 - \frac{s+1}{2s+1}) + 1 - \frac{s+1}{2s+1} = 1$.

3. UP preserves the satisfiability when applied to a simple Ł-clausal form φ. If UP(φ) contains the empty clause, then φ is unsatisfiable. If UP(φ) is the empty formula, then φ is satisfiable. In the rest of cases, since UP(φ) contains no unit clauses, the satisfiability of UP(φ) can be decided using either case 1 or case 2 of this lemma. □

Lemma 2 *The satisfiability of any simple* Ł*-clausal form is decidable in linear time.*

Proof Case 1 of Lemma 1 can be clearly solved in linear time because we only have to check whether the cardinality of the truth value set is either odd or even. In the latter case, we also have to check whether all the clauses contain at least three literals, and this can be achieved by traversing once the clausal form.

Case 2 of Lemma 1 can be also solved in linear time. Checking whether the cardinality of the truth value set is even, and identifying the binary clauses in the simple Ł-clausal form can be easily done in linear time. In addition, there are algorithms for solving the resulting Boolean 2-SAT problem in linear time (Aspvall et al. 1979).

Case 3 of Lemma 1 can be solved using the same algorithms that are applied for Boolean unit propagation, which have linear-time complexity (Zhang and Stickel 1996). □

3.2 Ł-clausal forms

To overcome the limitations of simple Ł-clausal forms explained above, we now define a new family of test instances, called **Łukasiewicz clausal forms (Ł-clausal forms)**. These instances have a higher expressive power and are interesting from a practical point of view because they exhibit an easy-hard-easy pattern and a phase transition phenomenon similar to the ones found in other combinatorial problems like Boolean 3-SAT (Mitchell et al. 1992) and regular 3-SAT (Béjar and Manyà 1999; Manyà et al. 1998). So, one can generate both satisfiable and unsatisfiable instances of varying difficulty by adjusting the clause-to-variable ratio.

Definition 5
- A ***literal*** is a propositional variable or a negated propositional variable.
- A ***term*** is either a literal or an expression of the form $\neg(l_1 \oplus \cdots \oplus l_n)$, where $l_1, \ldots, l_n$ are literals.
- A ***Łukasiewicz clause (Ł-clause)*** is an expression of the form $t_1 \oplus \cdots \oplus t_n$, where $t_1, \ldots, t_n$ are terms.
- A ***Łukasiewicz clausal form (Ł-clausal form)*** is a (weak) conjunction of Ł-clauses; i.e., it is an expression of the form $\bigwedge_{i=1}^{k} C_i$, where C_i is an Ł-clause.

Definition 6 The ***SAT*** problem for an Ł-clausal form consists in finding an interpretation that satisfies all its Ł-clauses. If each Ł-clause contains exactly k literals, it is called the k-SAT problem for Ł-clausal forms.

Example 2 The Ł-clausal form $\neg x_2 \wedge (x_1 \oplus x_3) \wedge (\neg(x_1 \oplus x_2) \oplus \neg x_3)$ is satisfied by the interpretation that assigns the value 0 to x_1 and x_2 and assigns the value 1 to x_3.

Lemma 3 *The 3-SAT problem for Ł-clausal forms is NP-complete.*

Proof We will show that (i) this problem belongs to NP and (ii) the Boolean 3-SAT problem is polynomially reducible to the 3-SAT problem for Ł-clausal forms.

The 3-SAT problem for Ł-clausal forms clearly belongs to NP: Given a satisfiable Ł-clausal form, a nondeterministic algorithm can guess a satisfying interpretation and check that it satisfies the formula in polynomial time.

Let $\{l_{ij} | 1 \leq i \leq n, 1 \leq j \leq 3\}$ be a set of literals over the set of Boolean variables $\{x_1, \ldots, x_m\}$, and let $\phi = \bigwedge_{i=1}^{n}(l_{i1} \vee l_{i2} \vee l_{i3})$ be a Boolean 3-SAT instance. We derive an Ł-clausal form 3-SAT instance ϕ' from ϕ as follows:

1. For every Boolean variable x_k, $1 \leq k \leq m$, we add the Ł-clause $\neg(x_k \oplus x_k) \oplus x_k$ in ϕ'.
2. For every clause $l_{i1} \vee l_{i2} \vee l_{i3}$ in ϕ, we add the Ł-clause $l_{i1} \oplus l_{i2} \oplus l_{i3}$ in ϕ'.

So, $\phi' = \bigwedge_{k=1}^{m}(\neg(x_k \oplus x_k) \oplus x_k) \wedge \bigwedge_{i=1}^{n}(l_{i1} \oplus l_{i2} \oplus l_{i3})$. This reduction can obviously be performed in polynomial time, and the size of ϕ' is linear in the size of ϕ.

We now prove that ϕ' is satisfiable iff ϕ is satisfiable. Assume that ϕ' is satisfiable. Then, every variable x_k must be evaluated to either 0 or 1. This is so because $\neg(x_k \oplus x_k) \oplus x_k$

evaluates to 1 iff x_k evaluates to either 0 or 1. Since the semantics of Łukasiewicz strong disjunction when restricted to 0 and 1 is identical to the semantics of Boolean disjunction, any model of ϕ' is a model of ϕ. Therefore, ϕ is satisfiable.

Assume that ϕ is satisfiable. Any Boolean model of ϕ can be transformed to a many-valued model by assigning 0 to the variables that evaluate to false, and 1 to the variables that evaluate to true. If x_k evaluates either to 0 or 1, $\neg(x_k \oplus x_k) \oplus x_k$ evaluates to 1, and $l_{i1} \oplus l_{i2} \oplus l_{i3}$ also evaluates to 1 because we assumed that every Boolean clause $l_{i1} \vee l_{i2} \vee l_{i3}$ is satisfied. Therefore, ϕ' is satisfiable. □

These instances are interesting because the 3-SAT problem for Ł-clausal forms is NP-complete while it is decidable in linear time if negations are not allowed above the literal level; i.e., the 3-SAT problem for simple Ł-clausal forms. Moreover, Ł-clausal forms are genuinely multiple-valued in the sense that there exist Ł-clausal forms that are satisfiable under Łukasiewicz semantics but are unsatisfiable under Boolean semantics.

Example 3 The formula $(x_1 \oplus x_2) \wedge (\neg x_1 \oplus x_2) \wedge (x_1 \oplus \neg x_2) \wedge (\neg x_1 \oplus \neg x_2)$ is satisfied in Łukasiewicz logic if x_1 and x_2 evaluate to $\frac{1}{2}$, but $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$ is unsatisfiable in the Boolean case.

We now prove that the 2-SAT problem for Ł-clausal forms can be solved in linear time.

Lemma 4 *The 2-SAT problem for Ł-clausal forms is decidable in linear time.*

Proof The only Ł-clauses of 2-SAT instances that are different from the clauses in simple Ł-clausal forms are of the form $\neg(l_i \oplus l_j)$, where l_i, l_j are literals. However, clauses of the form $\neg(l_i \oplus l_j)$ can be replaced with $\neg l_i \wedge \neg l_j$ because $\neg(l_i \oplus l_j)$ is satisfiable iff l_i and l_j evaluate both to 0. Hence, replacing all the Ł-clauses $\neg(l_i \oplus l_j)$ with $\neg l_i \wedge \neg l_j$ produces a simple Ł-clausal form, whose satisfiability can be decided in linear time according to Lemma 2. □

3.3 Restricted Ł-clausal forms

In the proof of Lemma 3, we saw that the full expressive capacity of Ł-clausal forms is not necessary for checking the NP-completeness of SAT over those formulas. Indeed, we can naturally provide an alternative clausal form for Łukasiewicz logic that still is NP-complete, while it is possibly simpler in some aspects than the Ł-clausal forms introduced above.

Definition 7 An Ł-clausal form is a ***restricted Ł-clausal form*** iff it is of the form

$$\bigwedge_{i \in I} \bigoplus_{j \in J_i} l_{ij}^{c_{ij}}$$

for I, J_i finite sets of indexes, l_{ij} a literal for each i, j in the corresponding indexing sets, and $c_{ij} \in \mathbb{N}$.

Example 4 The Ł-clausal form $\neg x_2 \wedge (\neg(x_1 \oplus x_1) \oplus x_3) \wedge (\neg(\neg x_2 \oplus \neg x_2) \oplus \neg x_3)$ corresponds to the restricted Ł-clausal form $\neg x_2 \wedge ((\neg x_1)^2 \oplus x_3) \wedge ((x_2)^2 \oplus \neg x_3)$.

Lemma 5 *The 3-SAT problem for restricted Ł-clausal forms is NP-complete.*

Proof This can be proved in an analogous way to Lemma 3. Just observe that the formula $\neg(x \oplus x) \oplus x$ (added to the Ł-clausal form that reduces Boolean 3-SAT) is in fact equivalent, by the definition of $\oplus$ in Ł, to $(\neg x)^2 \oplus x$, which is a restricted Ł-clause. Thus, the same proof of NP-completeness is valid for restricted Ł-clausal forms. □

Lemma 6 *The 2-SAT problem for restricted Ł-clausal form is decidable in linear time.*

Proof Similarly to the proof of Lemma 4, observe that the only clauses different from simple Ł-clauses are of the form l^2, which simply imposes $l = 1$, not adding any complexity to the solution search. □

4 Reducing Łukasiewicz formulas to equivalent Ł-clausal forms

In a similar way as any Boolean propositional formula can be reduced to a polynomial-size CNF formula that preserves satisfiability (see, e.g., Tseitin 1968), we describe how we can polynomially reduce any formula φ in Łukasiewicz logic (with variables $\mathcal{V}ar(\varphi)$) to a satisfiability preserving Ł-clausal form φ^* over $\mathcal{V}ar(\varphi)$ and a set of auxiliary variables that abbreviate more complex formulas. The relevance of the translation in this context comes from the fact that the formulas encoding the equivalence between the new variables and the corresponding complex formulas can also be written as Ł-clauses.

Let φ be a Łukasiewicz formula[3] and let $SFm(\varphi)$ denote the set of subformulas of φ. For each χ in $SFm(\varphi)$ that is not a literal nor φ, we introduce a new auxiliary variable x_χ. Then, let φ' be defined by substituting the formula(s) of the last level of operations by the corresponding (possibly new) variables. That is to say,

$$\begin{aligned} x' &:= x \\ (\neg\psi)' &:= \neg x_\psi \\ (\psi \rightarrow \chi)' &:= \neg x_\psi \oplus x_\chi \end{aligned}$$

[3] Recall that any Łukasiewicz formula can be written using only $\neg$ and $\rightarrow$ operators (see, e.g., Cignoli et al. 2000), and that $\varphi \rightarrow \psi \equiv \neg\varphi \oplus \psi$ in Łukasiewicz logic.

where x_ψ is simply ψ in the case ψ is a literal. Then, let $\Delta(\varphi)$ be the set of formulas inductively fixing the value of each new x_ψ to its intended one, by relying on the equivalence of $\psi \to \chi$ and $\neg\psi \oplus \chi$ in Łukasiewicz logic. Formally, for each new variable $x_{\neg\psi}$, add the formulas

$$\neg x_{\neg\psi} \oplus \neg x_\psi \quad \text{and} \quad x_{\neg\psi} \oplus x_\psi$$

and, for each new variable $x_{\psi\to\chi}$, add the formulas

$$\neg x_{\psi\to\chi} \oplus \neg x_\psi \oplus x_\chi \quad \text{and} \quad \neg(\neg x_\psi \oplus x_\chi) \oplus x_{\psi\to\chi}$$

Example 5 Consider the formula

$$\varphi \equiv ((z \to w) \to \neg z) \to w.$$

Then, $\varphi' \equiv \neg x_{((z\to w)\to\neg z)} \oplus w$, and

$$\begin{aligned}\Delta(\varphi) = \{&\neg x_{(z\to w)\to\neg z} \oplus \neg x_{z\to w} \oplus \neg z,\\ &\neg(\neg x_{z\to w} \oplus \neg z) \oplus x_{((z\to w)\to\neg z)}\\ &\neg x_{z\to w} \oplus \neg z \oplus w\\ &\neg(\neg z \oplus w) \oplus x_{z\to w}\}\end{aligned}$$

Let $\varphi^* := \varphi' \wedge \bigwedge_{\chi\in\Delta(\varphi)} \chi$. Observe that the total number of conjuncts of φ^* is bounded by $1+2(|SFm(\varphi)|-1-|\mathcal{V}(\varphi)|)$.

Proposition 1 *Let φ be a Łukasiewicz formula and let φ^* be the formula obtained with the above-defined translation. Then,*

1. *φ^* is a Ł-clausal form with at most three literals in each clause.*
2. *For each interpretation, I into $\mathbf{Ł}_\infty$, if $I(\Delta(\varphi)) \subseteq \{1\}$, then*

$$I(\varphi) = I(\varphi').$$

Proof 1. By definition, φ' is a clause of a simple Ł-clausal form (and thus, a clause of a Ł-clausal form), that has at most two variables (one if $\varphi \equiv \neg\psi$ for some ψ). Also following the definition, each one of the formulas in $\Delta(\varphi)$ is a clause in Ł-clausal form with at most three literals (three for the clauses encoding the value of a variable of the form $x_{\psi\to\chi}$, and two for the ones concerning $x_{\neg\psi}$).

2. We can prove by induction on φ that $I(\Delta(\varphi)) \subseteq \{1\}$ implies that $I(\varphi) = I(\varphi')$. Observe before proceeding that if ψ is a subformula of φ, then $I(\Delta(\varphi)) \subseteq \{1\}$ implies $I(\Delta(\psi)) \subseteq \{1\}$.
   - For propositional variables it is trivial, because φ and φ' are the same formula.
   - Assume $\varphi \equiv \neg\psi$. By definition, $I(\varphi') = I(\neg x_\psi)$. Since $I(\Delta(\varphi)) \subseteq \{1\}$, in particular it holds that $I(\neg x_\psi \oplus \psi') = 1$ and $I(\neg\psi' \oplus x_\psi) = 1$. By interpretation of the Ł connectives, we get that $I(x_\psi) = I(\psi')$, and so $I(\varphi') = I(\neg\psi') = \neg I(\psi')$. Now, from the observation above, we can apply the induction hypothesis and get $\neg I(\psi') = \neg I(\psi)$, and so conclude $I(\varphi') = I(\neg\psi) = I(\varphi)$.
   - If $\varphi \equiv \psi_1 \to \psi_2$, so $I(\varphi') = I(\neg x_{\psi_1} \oplus \psi_2')$. As before, it is immediate that $I(\varphi') = \neg I(\psi_1') \oplus I(\psi_2')$. By the induction hypothesis, this equals to $\neg I(\psi_1) \oplus I(\psi_2) = I(\varphi)$, concluding the proof.

□

An immediate corollary of the previous translation, together with Lemma 3, is a new proof of the already known result of NP-completeness of SAT in Łukasiewicz logic (e.g., Mundici 1987).

It seems, however, not clear whether it is possible to do a reduction from arbitrary Łukasiewicz formulas to Ł-clausal forms preserving the set of variables and allowing longer clauses (which is easy in classical logic although the derived formula can have exponential size). In contrast to what happens in classical logic (with the lattice operators), in Łukasiewicz logic $\odot$ does not distribute over $\oplus$.

5 Experimental results

We first describe the generator of Ł-clausal forms that we have developed and then report on the empirical investigation conducted to identify challenging benchmarks.

The generator of Ł-clausal form 3-SAT instances used works as follows: given n variables and k clauses, each of the k clauses is constructed from three variables $(x_{i_1}, x_{i_2}, x_{i_3})$ which are drawn uniformly at random. Then, one of the following eleven possible Ł-clauses $x_{i_1} \oplus x_{i_2} \oplus x_{i_3}$, $\neg x_{i_1} \oplus x_{i_2} \oplus x_{i_3}$, $x_{i_1} \oplus \neg x_{i_2} \oplus x_{i_3}$, $x_{i_1} \oplus x_{i_2} \oplus \neg x_{i_3}$, $\neg x_{i_1} \oplus \neg x_{i_2} \oplus x_{i_3}$, $\neg x_{i_1} \oplus x_{i_2} \oplus \neg x_{i_3}$, $x_{i_1} \oplus \neg x_{i_2} \oplus \neg x_{i_3}$, $\neg x_{i_1} \oplus \neg x_{i_2} \oplus \neg x_{i_3}$, $\neg(x_{i_1} \oplus x_{i_2}) \oplus x_{i_3}$, $\neg(x_{i_1} \oplus x_{i_3}) \oplus x_{i_2}$, and $x_{i_1} \oplus \neg(x_{i_2} \oplus x_{i_3})$ is selected with the same probability. We consider this set of clauses because it can be observed in the reduction from Boolean 3-SAT to Ł-clausal forms that these types of negations suffice to get NP-hardness.

In the experiments, we solved sets of 100 Ł-clausal form 3-SAT instances with 1500 variables, and a number of variables ranging from 100 to 6000 with steps of 100. We considered the 3-valued, 5-valued and 7-valued Łukasiewicz logics, as well as the infinitely-valued Łukasiewicz logic. Instances were solved with the SMT solver Yices (Dutertre 2014) (version 2.5.1), with a theorem prover similar to those described in Ansótegui et al. (2012), as well as with the exact MIP solver SCIP (Cook et al. 2013) (version 3.0.0) linked with the linear programming solver CPLEX (version 12.6).

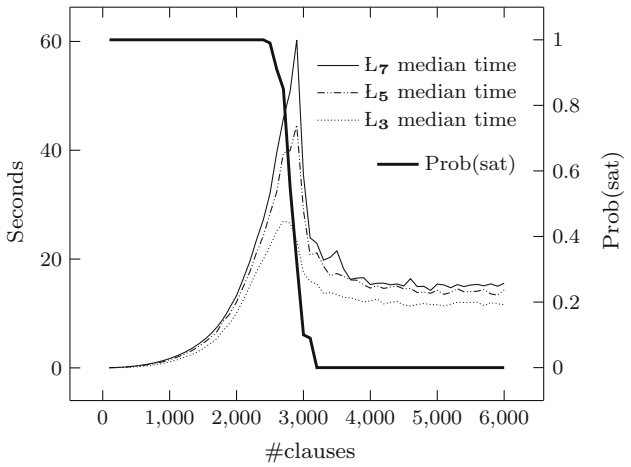

Fig. 1 Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in 3-valued, 5-valued and 7-valued Łukasiewicz logics (Yices 2.5.1)

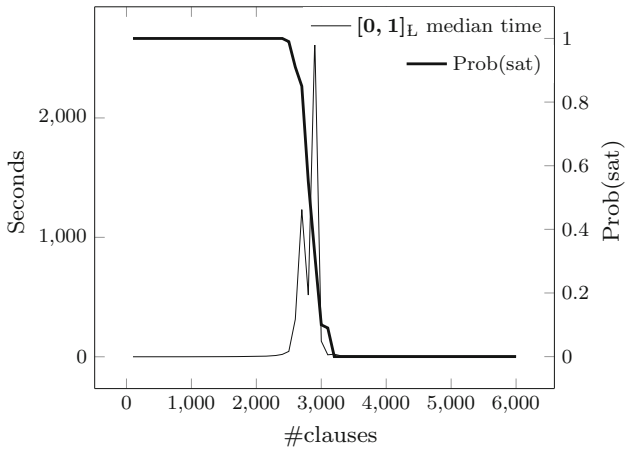

Fig. 2 Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in infinitely-valued Łukasiewicz logic (Yices 2.5.1)

The experiments were run on an Intel® Xeon® E3-1220v2 machine at 3.10 GHz with Turbo Boost disabled.

We start by depicting the results obtained with the SMT solver. Figure 1 shows the results for the finitely-valued case. We observe a phase transition between satisfiability and unsatisfiability similar to that of Boolean random 3-SAT, as well as an easy-hard-easy pattern in the median difficulty of the problems around the phase transition. Prob(sat) indicates the probability that an instance has to be satisfiable. In the threshold point of the phase transition, roughly half of the instances are satisfiable; on its left, most of the instances are satisfiable; and on its right, most of the instances are unsatisfiable. Moreover, we observe that the difficulty increases with the cardinality of the truth value set, especially in the hardest instances.

It is worth noting that in Crawford and Auton (1993) the threshold point for Boolean random 3-SAT was accurately identified to correspond to a clause-to-variable ratio equal to 4.24. Interestingly, in our case it corresponds to a ratio of approximately 1.9, regardless of the cardinality of the truth value set.

Figure 2 shows the results for the infinitely-valued case. We observe the same phase transition and a similar easy-hard-easy pattern, but the hard instances become pretty much harder, with median solving times increasing to thousands of seconds.

We have also considered solving the generated formulas with a mixed integer programming (MIP) solver; in particular, using the MIP solver SCIP, which allows to solve MIPs exactly over the rational numbers. Note that all standard MIP solvers work with finite precision (floating point) arithmetic. This allows efficient computations but also introduces rounding errors, which cannot be neglected in our setting, as they could lead to incorrect results. On the other side, all SMT solvers, as well as the exact version of the MIP solver SCIP, work with arbitrary precision arithmetic, typically using the GNU Multiple Precision Arithmetic Library (GMP).

For the MIP experiments, we use the encodings to MIP defined in Hähnle (1994). They are summarized in the following lemmas.

Lemma 7 *For any infinitely-valued Ł-clausal forms ϕ and ψ, any $i \in [0, 1]$ and any interpretation I,*

1. $I(\neg\phi) \geq i$ *iff* $I(\phi) \leq 1 - i$.
2. $I(\neg\phi) \leq i$ *iff* $I(\phi) \geq 1 - i$.
3. $I(\phi \oplus \psi) \geq i$ *iff there are* $i_1, i_2 \in [0, 1]$ *such that*

$$I(\phi) \geq i_1$$
$$I(\psi) \geq i_2$$
$$i_1 + i_2 = i$$

4. $I(\phi \oplus \psi) \leq i$ *iff there are* $i_1, i_2 \in [0, 1]$ *and* $v_{\text{aux}} \in \{0, 1\}$ *such that*

$$I(\phi) \leq i_1$$
$$yI(\psi) \leq i_2$$
$$v_{\text{aux}} \leq i$$
$$v_{\text{aux}} \leq i_1$$
$$v_{\text{aux}} \leq i_2$$
$$v_{\text{aux}} + i = i_1 + i_2$$

Note that $I(\phi \oplus \psi) \leq i$ will hold either if $I(\phi) + I(\psi) \leq i$ or $i = 1$. If $v_{\text{aux}} = 0$, then the constraint system simplifies to $I(\phi) \leq i_1, I(\psi) \leq i_2, 0 \leq i, 0 \leq i_1, 0 \leq i_2, i = i_1 + i_2$, where $0 \leq i, 0 \leq i_1$, and $0 \leq i_2$ are redundant. If $v_{\text{aux}} = 1$, then the constraint system simplifies to $I(\phi) \leq i_1, I(\psi) \leq i_2, 1 \leq i, 1 \leq i_1, 1 \leq i_2, 1 + i = i_1 + i_2$, which implies $i = 1, i_1 = 1, i_2 = 1$, making the system to become trivially satisfiable.

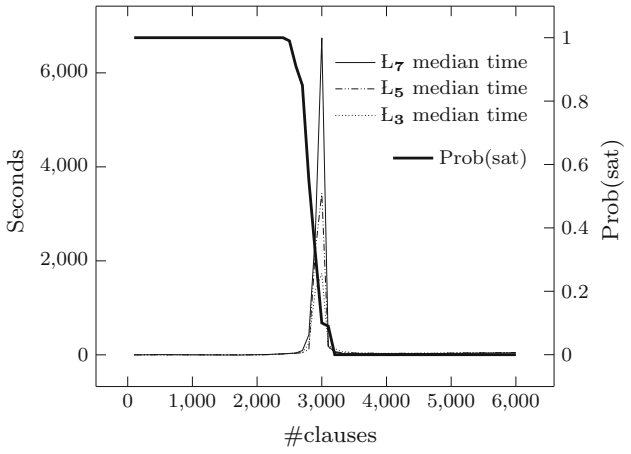

Fig. 3 Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in 3-valued, 5-valued and 7-valued Łukasiewicz logics (SCIP 3.0.0)

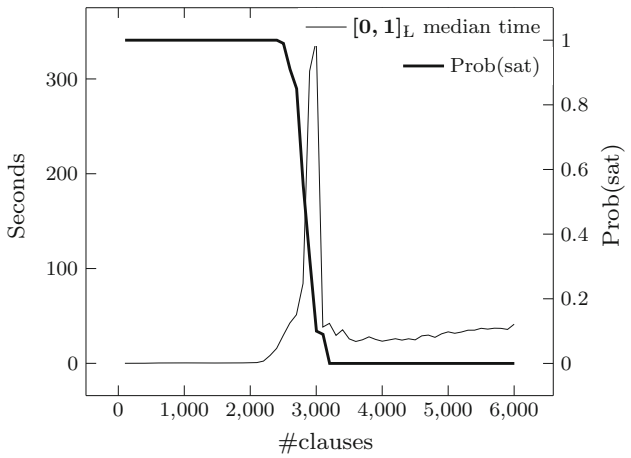

Fig. 4 Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in infinitely-valued Łukasiewicz logic (SCIP 3.0.0)

We consider now the finitely-valued case with $N = \{0, 1, \ldots, n-1\}$ and $D = \{n-1\}$.[4]

Lemma 8 *For any finitely-valued Ł-clausal forms ϕ and ψ, any $i \in \{0, 1, \ldots, n-1\}$ and any interpretation I,*

1. $I(\neg\phi) \geq i$ *iff* $I(\phi) \leq n-1-i$.
2. $I(\neg\phi) \leq i$ *iff* $I(\phi) \geq n-1-i$.
3. $I(\phi \oplus \psi) \geq i$ *iff there are* $i_1, i_2 \in \{0, 1, \ldots, n-1\}$ *such that*

$$
\begin{aligned}
&yI(\phi) \geq i_1\\
&I(\psi) \geq i_2\\
&i_1 + i_2 = i
\end{aligned}
$$

4. $I(\phi \oplus \psi) \leq i$ *iff there are* $i_1, i_2 \in \{0, 1, \ldots, n-1\}$ *and* $v_{\text{aux}} \in \{0, 1\}$ *such that*

$$
\begin{aligned}
&I(\phi) \leq i_1\\
&I(\psi) \leq i_2\\
&(n-1) \times v_{\text{aux}} \leq i\\
&(n-1) \times v_{\text{aux}} \leq i_1\\
&y(n-1) \times v_{\text{aux}} \leq i_2\\
&(n-1) \times v_{\text{aux}} + i = i_1 + i_2
\end{aligned}
$$

Figure 3 shows the results for the finitely-valued case when using the exact version of the MIP solver SCIP with the encoding described in Lemma 8. We observe the same pattern as in Fig. 1 for the results with the SMT solver Yices, but here the solving times are about two orders of magnitude higher in the hardest instances.

[4] The set $\{0, 1, \ldots, n-1\}$ is isomorphic to the set $\{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$ through the isomorphism $i_n : \{0, 1, \ldots, n-1\} \to \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$, $i_n(x) = \frac{x}{n-1}$.

Figure 4 shows the results for the infinitely-valued case when using the exact version of the MIP solver SCIP with the encoding described in Lemma 7. In this case, the MIP solver exhibits much better performance in the hardest instances, unlike the SMT solver.

We have identified the same phase transition phenomenon and easy-hard-easy pattern for Ł-clausal form 3-SAT instances also with other SMT solvers, as well as with other MIP solvers. Hence, it becomes apparent that our results are independent from the encoding and the solver used and provide a challenging benchmark. The results also suggest that the SMT-based approach is more suitable in the finitely-valued case, whereas the MIP-based approach is more suitable in the infinitely-valued case.

6 Concluding remarks

We have defined three new clausal forms for Łukasiewicz logic (simple Ł-clausal forms, Ł-clausal forms and restricted Ł-clausal forms), analyzed the complexity of different satisfiability problems for these clausal forms, and proposed a method for translating any Łukasiewicz formula into a satisfiability preserving Ł-clausal form. We have also described a generator that produces Ł-clausal forms of varying difficulty to test Łukasiewicz SAT solvers and conducted an empirical investigation with SMT and MIP solvers to identify an easy-hard-easy pattern and a phase transition phenomenon. As future work, we plan to analytically derive tight lower and upper bounds of the threshold point and find suitable encodings of combinatorial problems using the formalism of Ł-clausal forms.

Finally, we would like to mention the existence of two related works (Borgwardt et al. 2014; Bofill et al. 2015a)

that show the NP-completeness of the satisfiability problem of Łukasiewicz rules, which are a fragment of Ł-clausal forms. Borgwardt et al. (2014) proved the NP-completeness of Łukasiewicz rules when the cardinality of the truth value set is greater than three. We then proved in Bofill et al. (2015a) that the problem remains NP-complete when the truth value set has three elements, and that it can be solved in polynomial time when the rules have almost one literal in the consequent of the rule.

Acknowledgements The authors would like to thank the anonymous reviewer for their valuable comments and suggestions. This work was supported by Project LOGISTAR from the EU H2020 Research and Innovation Programme under Grant Agreement No. 769142, MINECO-FEDER Projects RASO (TIN2015-71799-C2-1-P) and LoCos (TIN2015-66293-R), and the UdG Project MPCUdG2016/055. Vidal acknowledges the support from the Project GA17-04630S of the Czech Science Foundation (GAČR).

Compliance with ethical standards

Conflict of interest Authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Aguzzoli S, Gerla B, Haniková Z (2005) Complexity issues in basic logic. Soft Comput 9(12):919–934

Ansótegui C, Bofill M, Manyà F, Villaret M (2012) Building automated theorem provers for infinitely-valued logics with satisfiability modulo theory solvers. In: Proceedings, 42nd international symposium on multiple-valued logics (ISMVL), Victoria, BC, Canada. IEEE CS Press, pp 25–30

Ansótegui C, Bofill M, Manyà F, Villaret M (2015) SAT and SMT technology for many-valued logics. Mult Valued Logic Soft Comput 24(1–4):151–172

Ansótegui C, Bofill M, Manyà F, Villaret M (2016) Automated theorem provers for multiple-valued logics with satisfiability modulo theory solvers. Fuzzy Sets Syst 292:32–48

Aspvall R, Plass M, Tarjan R (1979) A linear time algorithm for testing the truth of certain quantified boolean formulae. Inf Process Lett 8(3):121–123

Beckert B, Hähnle R, Manyà F (1999) Transformations between signed and classical clause logic. In: Proceedings, international symposium on multiple-valued logics, ISMVL'99, Freiburg, Germany. IEEE Press, Los Alamitos, pp 248–255

Beckert B, Hähnle R, Manyà F (2000) The 2-SAT problem of regular signed CNF formulas. In: Proceedings. 30th international symposium on multiple-valued logics (ISMVL), Portland/OR, USA. IEEE CS Press, Los Alamitos, pp 331–336

Béjar R, Manyà F (1999) Phase transitions in the regular random 3-SAT problem. In: Proceedings of the 11th international symposium on methodologies for intelligent systems, ISMIS'99, Warsaw, Poland. Springer LNAI 1609, pp 292–300

Bofill M, Manyà F, Vidal A, Villaret M (2015a) The complexity of 3-valued łukasiewicz rules. In: Proceedings, 12th international conference on modeling decisions for artificial intelligence, MDAI, Skövde, Sweden. Springer LNCS 9321, pp 221–229

Bofill M, Manyà F, Vidal A, Villaret M (2015b) Finding hard instances of satisfiability in Łukasiewicz logics. In: Proceedings, 45th international symposium on multiple-valued logics (ISMVL), Waterloo, Canada. IEEE CS Press, pp 30–35

Borgwardt S, Cerami M, Peñaloza R (2014) Many-valued Horn logic is hard. In: Proceedings of the first workshop on logics for reasoning about preferences, uncertainty, and vagueness, PRUV 2014, co-located with IJCAR 2014, Vienna, Austria, pp 52–58

Cignoli R, D'Ottaviano IML, Mundici D (2000) Algebraic foundations of many-valued reasoning, trends in logic-studia logica library, vol 7. Kluwer Academic Publishers, Dordrecht

Cintula P, Hájek P, Noguera C (eds) (2011) Handbook of mathematical fuzzy logic, 3 volumes, studies in logic. mathematical logic and foundation, vols 37, 38 and 58. College Publications

Cook W, Koch T, Steffy DE, Wolter K (2013) A hybrid branch-and-bound approach for exact rational mixed-integer programming. Math Program Comput 5(3):305–344

Crawford JM, Auton LD (1993) Experimental results on the crossover point in satisfiability problems. In: Proceedings of the 11th national conference on artificial intelligence, AAAI'93, Washington, DC, USA. AAAI Press, pp 21–27

Davis M, Logemann G, Loveland D (1962) A machine program for theorem-proving. Commun ACM 5:394–397

Dutertre B (2014) Yices 2.2. In: Proceedings of the 26th international conference on computer aided verification, CAV 2014, Lecture Notes in Computer Science, vol 8559. Springer, pp 737–744

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco

Hähnle R (1994) Short conjunctive normal forms in finitely-valued logics. J Log Comput 4(6):905–927

Hájek P (1998) Metamathematics of fuzzy logic. Kluwer, Dordrecht

Li CM, Manyà F (2009) MaxSAT, hard and soft constraints. In: Biere A, van Maaren H, Walsh T (eds) Handbook of satisfiability. IOS Press, Amsterdam, pp 613–631

Manyà F (2000) The 2-SAT problem in signed CNF formulas. Mult Valued Log Int J 5(4):307–325

Manyà F, Béjar R, Escalada-Imaz G (1998) The satisfiability problem in regular CNF-formulas. Soft Comput 2(3):116–123

Metcalfe G, Olivetti N, Gabbay D (2005) Lukasiewicz logic: From proof systems to logic programming. Log J IGPL 12(5):561–585

Metcalfe G, Olivetti N, Gabbay DM (2009) Proof theory of fuzzy logics, applied logic series, vol 36. Springer, Berlin

Mitchell D, Selman B, Levesque H (1992) Hard and easy distributions of SAT problems. In: Proceedings of the 10th national conference on artificial intelligence, AAAI'92, San Jose, CA, USA. AAAI Press, pp 459–465

Mundici D (1987) Satisfiability in many-valued sentential logic is NP-complete. Theor Comput Sci 52:145–153

Mundici D, Olivetti N (1998) Resolution and model building in the infinitely-valued calculus of Łukasiewicz. Theor Comput Sci 200(1–2):335–366

Tseitin G (1968) Studies in constructive mathematics and mathematical logic, part II, chap. In: On the complexity of derivations in the propositional calculus. Steklov Mathematical Institute, pp 115–125

Vidal A (2016) MNiBLoS: a SMT-based solver for continuous t-norm based logics and some of their modal expansions. Inf Sci 372:709–730

Vidal A, Bou F, Godo L (2012) An SMT-based solver for continuous t-norm based logics. In: Proceedings of the 6th international conference on scalable uncertainty management, SUM 2012, Marburg, Germany. Springer LNCS 7520, pp 633–640

Wagner H (1998) A new resolution calculus for the infinite-valued propositional logic of lukasiewicz. In: FTP (LNCS selection). Technical Report E1852-GS-981, TU Wien, pp 234–243

Zhang H, Stickel ME (1996) An efficient algorithm for unit propagation. In: In Proceedings of the fourth international symposium on artificial intelligence and mathematics (AI-MATH'96), Fort Lauderdale (Florida), USA, pp 166–169