Finding Hard Instances of Satisfiability in Łukasiewicz Logics[*]

Miquel Bofill
UdG
Girona, Spain

Felip Manyà
IIIA–CSIC
Bellaterra, Spain

Amanda Vidal
IIIA–CSIC
Bellaterra, Spain

Mateu Villaret
UdG
Girona, Spain

Abstract

One aspect that has been poorly studied in multiple-valued logics, and in particular in Łukasiewicz logics, is the generation of instances of varying difficulty for evaluating, comparing and improving satisfiability solvers. In this paper we present a new class of clausal forms, called Łukasiewicz (Ł-)clausal forms, motivate their usefulness, study their complexity, and report on an empirical investigation that shows an easy-hard-easy pattern and a phase transition phenomenon when testing the satisfiability of Ł-clausal forms.

1 Introduction

The proof theory of multiple-valued logics, as well as its complexity, have been deeply studied for a wide variety of logics [1, 10, 11]. However, the development of satisfiability solvers has received less attention despite of 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 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 [2, 3, 13, 14], as well as the need of empirically evaluating and comparing them with other existing approaches, the objective of this paper is to develop a suitable benchmark for satisfiability testing in Łukasiewicz logics. The ultimate goal is to devise a random generator of instances of varying difficulty.

More specifically, this paper starts by analyzing how the Conjunctive Normal Forms (CNFs) used by SAT solvers can be extended to Łukasiewicz logics. In a first attempt, we replace the classical disjunction in Boolean CNFs with Łukasiewicz strong disjunction, and interpret negation using Łukasiewicz negation. Curiously, 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 is greater than two). This result is surprising because deciding the satisfiability of Boolean CNFs is NP-complete when there are clauses with at least three literals [9].

To produce computationally difficult instances we then define 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.[1] Hence, these problems have the same complexity as their Boolean counterparts.

Moreover, we report on an empirical investigation in which we identify —when testing the satisfiability of Ł-clausal forms having a fixed number of literals per clause, and generated uniformly at random— 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 a generator that is able to produce both satisfiable and unsatisfiable instances of varying difficulty.

The paper is structured as follows. Section 2 defines ba-

[*]Research partially supported by the Generalitat de Catalunya grant AGAUR 2014-SGR-118, and the Ministerio de Economía y Competividad projects AT CONSOLIDER CSD2007-0022, INGENIO 2010, CO-PRIVACY TIN2011-27076-C03-03, EDETRI TIN2012-39348-C02-01 and HeLo TIN2012-33042. The second author was supported by Mobility Grant PRX14/00195 of the Ministerio de Educación, Cultura y Deporte.

[1]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.

0195-623X/15 $31.00 © 2015 IEEE
DOI 10.1109/ISMVL.2015.10

sic concepts in Łukasiewicz logics. Section 3 defines two 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 second type. Section 4 reports on an empirical investigation that shows an easy-hard-easy pattern and a phase transition phenomenon when testing the satisfiability of the second type of clausal forms. Section 5 concludes and points out future research directions.

2 Preliminaries

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

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 triplet $\langle \mathbb{L}, N, A \rangle$ where $\mathbb{L} =< \Theta, \alpha >$ is a propositional language, N is a truth value set, and A is an interpretation of the operation symbols that assigns to each $\theta \in \Theta$ a function $A_\theta : N^{\alpha(\theta)} \rightarrow N$.

Many-valued logics are equipped with a non-empty subset D of N called the designated truth values, which are the truth values that are considered to affirm satisfiability.

Definition 3. Let $\mathcal{L}$ be a many-valued logic. An **interpretation on** $\mathcal{L}$ is a function $I : \mathcal{V} \rightarrow N$. 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 such that $I(\phi) \in D$.

Through this work, we focus in 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 20th century, and have been deeply studied both from theoretical and practical points of view. For a deeper study on these matters, see for instance [10].

The language of Łukasiewicz logic is given by

$$\mathbb{L}_{\text{Ł}uk} =< \neg/1, \rightarrow/2, \wedge/2, \vee/2, \odot/2, \oplus/2 > .$$

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.

Definition 4. The **infinitely-valued Łukasiewicz logic**, denoted by $\mathbf{[0,1]_{\text{Ł}}}$, is the many-valued logic $\langle \mathbb{L}_{\text{Ł}uk}, N, A_{\text{Ł}} \rangle$ equipped with the set of designated values $D = \{1\}$, where N is the real unit interval $[0, 1]$, and the interpretation of the operation symbols $A_{\text{Ł}}$ 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, $\mathbf{Ł_n} = \langle \mathbb{L}_{\text{Ł}uk}, N_n, A_{\text{Ł}} \rangle$ equipped with $D = \{1\}$. Note that the operations are well defined because N_n is a subalgebra of $[0, 1]$ with the interpretation of the operation symbols $A_{\text{Ł}}$ (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 $\mathbf{[0,1]_{\text{Ł}}}$ or $\mathbf{Ł_n}$ for some natural number n.

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

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

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

It is worth mentioning that one of the reasons we focus on Łukasiewicz logics is because $SAT^{Bool} \subset SAT^{\mathcal{L}}$. This is not the case for other relevant logics such as Gödel (G) and Product (Π), where $SAT^G = SAT^\Pi = SAT^{Bool}$ [10]. 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 [7], 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.

At first sight, one could generalize Boolean CNF formulas by replacing classical disjunction with Łukasiewicz strong disjunction, and negation with Łukasiewicz negation:

$$\bigwedge_{1\leq i\leq n} \Big(\bigoplus_{1\leq j\leq k_i} l_j^i \Big)$$

for $i, j, k_i, n \in \mathbb{N}$ and l_j^i literals. 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 shorter clause in a simple Ł-clausal form φ. That is to say, if $\varphi = \bigwedge_{1\leq i\leq n}(\bigoplus_{1\leq j\leq k_i} l_j^i)$, then $mc(\varphi) = \min\{k_i : 1 \leq i \leq n\}$. Through 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 6 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 5) 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 subformula by $B_2(\varphi)$; i.e., if $\varphi = \bigwedge_{1\leq i\leq n} C_i = \bigwedge_{1\leq i\leq n}(\bigoplus_{1\leq j\leq k_i} l_j^i)$, then

$$B_2(\varphi) = \bigwedge_{C_i\in\varphi, k_i=2} (l_1^i \vee l_2^i)$$

For example, 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 $SAT^{Bool} \subset SAT^{\mathcal{L}}$.

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 remove 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 $\perp$. We denote by $UP(\varphi)$ the formula obtained after applying UP to φ. $UP(\varphi)$ is either empty (meaning that it is satisfiable), a formula containing the bottom element $\perp$ (meaning that there exists a contradiction at the unitary level, and so, the formula is directly unsatisfiable) or $mc(UP(\varphi)) > 1$, and so we are in one of the scenarios discussed above.

Formally, we can express the previous results as follows.

Lemma 5. *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 $UP(\varphi)$ 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 $l_1 \oplus ... \oplus l_k$ 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 it 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 $UP(\varphi)$ contains the empty clause, then φ is unsatisfiable. If $UP(\varphi)$ is the empty formula, then φ is satisfiable. In the rest of cases, since $UP(\varphi)$ contains no unit clauses, the satisfiability of $UP(\varphi)$ can be decided using either case 1 or case 2 of this lemma.

Lemma 6. *The satisfiability of any simple Ł-clausal form is decidable in linear time.*

Proof. Case 1 of Lemma 5 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 5 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 [4].

Case 3 of Lemma 5 can be solved using the same algorithms that are applied for Boolean unit propagation, which have linear-time complexity [15].

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 [12]. So, one can generate both satisfiable and unsatisfiable instances of varying difficulty by adjusting the clause-to-variable ratio.

Definition 7. Let $\{x_1, \ldots, x_m\}$ be a set of propositional variables. A literal is a propositional variable x_i or a negated propositional variable $\neg x_i$. A term is a literal or an expression of the form $\neg(l_1 \oplus ... \oplus l_n)$, where $l_1, ..., l_n$ are literals. A Łukasiewicz clause (Ł-clause) is disjunction of terms. A Łukasiewicz clausal form (Ł-clausal form) is a conjunction of Ł-clauses.

Definition 8. 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 variable occurrences, it is called the k-SAT problem for Ł-clausal forms.

For example, 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 9. *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 $\{l_i^j | 1 \leq i \leq n, 1 \leq j \leq 3\}$ be a set of literals over the set of Boolean variables $\{x_1, \ldots, x_m\}$, and let $\phi = \bigwedge_{i=1}^{n}(l_i^1 \vee l_i^2 \vee l_i^3)$ 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 $l_i^1 \vee l_i^2 \vee l_i^3$ in ϕ, we add the Ł-clause $l_i^1 \oplus l_i^2 \oplus l_i^3$ in ϕ'.

So, $\phi' = \bigwedge_{k=1}^{m}(\neg(x_k \oplus x_k) \oplus x_k) \wedge \bigwedge_{i=1}^{n}(l_i^1 \oplus l_i^2 \oplus l_i^3)$. 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 $l_i^1 \oplus l_i^2 \oplus l_i^3$ also evaluates to 1 because we assumed that every Boolean clause $l_i^1 \vee l_i^2 \vee l_i^3$ 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. For example, $(x_1 \oplus x_2) \wedge (\neg x_1 \oplus x_2) \wedge (x_1 \oplus \neg x_2) \wedge (\neg x_1 \oplus \neg x_2)$ 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 10. *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 6.

4 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.

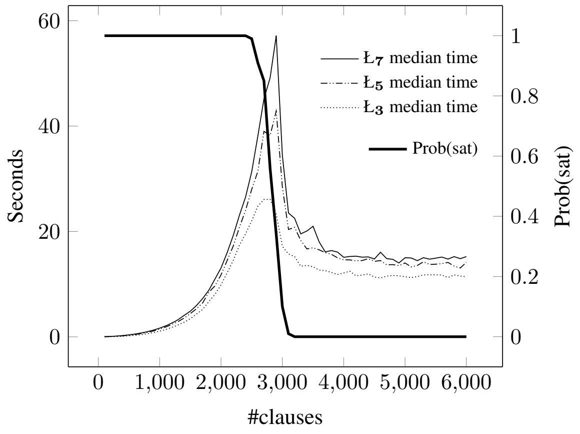

Figure 1. Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in 3-valued, 5-valued and 7-valued Łukasiewicz logics.

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 1,500 variables, and a number of variables ranging from 100 to 6,000 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 [8] (version 2.2.2), with a theorem prover similar to those described in [2]. The experiments were run on a cluster of machines, running the CentOS operating System, equipped with Intel® Xeon® E3-1220v2 Processors at 3.10 GHz with Turbo Boost disabled, and 8GB of main memory.

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 instance are satisfiable, and on its right most of the instance

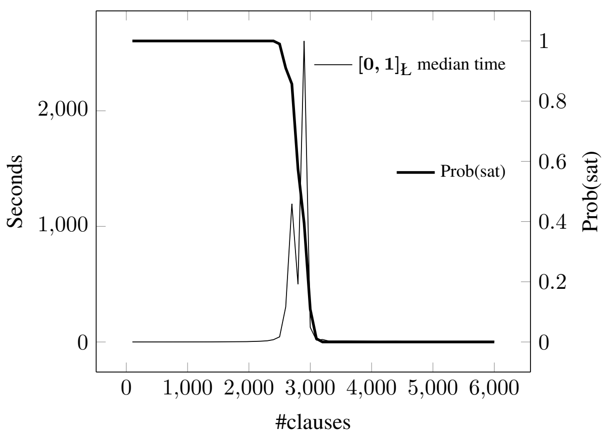

Figure 2. Phase transition and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with 1500 variables, in infinitely-valued Łukasiewicz logic. The number of variables is fixed to 1500.

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 [6] 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 a phase transition with a threshold point again of approximately 1.9, and a similar easy-hard-easy pattern.

Finally, it is worth mentioning that we have identified the phase transition phenomenon and easy-hard-easy pattern for Ł-clausal form 3-SAT instances with other SMT solvers, as well as with a MIP solver in the infinitely-valued case. Hence, our results are independent from the encoding and the solver used, and provide a challenging benchmark.

5 Concluding Remarks

We have defined a new class of clausal forms for Łukasiewicz logics, studied their complexity, and developed a generator that produces both satisfiable and unsatisfiable instances of varying difficulty. 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 that after the submission of our work we knew that recently, and independently of our work, Borgwardt et al. [5] proved that the satisfiability problem of a subclass of finitely-valued Łukasiewicz formulas is NP-complete when the cardinality of the truth value set is greater than three.

References

[1] S. Aguzzoli, B. Gerla, and Z. Haniková. Complexity issues in Basic logic. *Soft Computing*, 9(12):919–934, 2005.

[2] C. Ansótegui, M. Bofill, F. Manyà, and M. Villaret. 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*, pages 25–30. IEEE CS Press, 2012.

[3] C. Ansótegui, M. Bofill, F. Manyà, and M. Villaret. SAT and SMT technology for many-valued logics. *Multiple-Valued Logic and Soft Computing*, 24(1-4):151–172, 2015.

[4] R. Aspvall, M. Plass, and R. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulae. *Information Processing Letters*, 8(3):121–123, 1979.

[5] S. Borgwardt, M. Cerami, and R. Peñaloza. 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*, pages 52–58, 2014.

[6] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiability problems. In *11th National Conference on Artificial Intelligence, AAAI-93, Washington, DC, USA*, pages 21–27. AAAI Press / The MIT Press, 1993.

[7] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[8] B. Dutertre. Yices 2.2. In *Proceedings of the 26th International Conference on Computer Aided Verification, CAV 2014*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, 2014.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

[10] P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer, Dordrecht, 1998.

[11] G. Metcalfe, N. Olivetti, and D. M. Gabbay. *Proof Theory of Fuzzy Logics*, volume 36 of *Applied Logic Series*. Springer, 2009.

[12] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA*, pages 459–465. AAAI Press, 1992.

[13] A. Vidal. NiBLoS: a nice BL-logics solver. Master's thesis, Universitat de Barcelona, Barcelona, Spain, 2012.

[14] A. Vidal, F. Bou, and L. Godo. 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*, pages 633–640. Springer LNCS 7520, 2012.

[15] H. Zhang and M. E. Stickel. An efficient algorithm for unit propagation. In *In Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics (AI-MATH96), Fort Lauderdale (Florida USA*, pages 166–169, 1996.