Building Automated Theorem Provers for Infinitely-Valued Logics with Satisfiability Modulo Theory Solvers[*]

Carlos Ansótegui
Universitat de Lleida
Lleida, Spain

Miquel Bofill
Universitat de Girona
Girona, Spain

Felip Manyà
IIIA, CSIC
Bellaterra, Spain

Mateu Villaret
Universitat de Girona
Girona, Spain

Abstract

There is a relatively large number of papers dealing with complexity and proof theory issues of infinitely-valued logics. Nevertheless, little attention has been paid so far to the development of efficient solvers for such logics. In this paper we show how the technology of Satisfiability Modulo Theories (SMT) can be used to build efficient automated theorem provers for relevant infinitely-valued logics, including Łukasiewicz, Gödel and Product logics. Moreover, we define a test suite for those logics, and report on an experimental investigation that evaluates the practical complexity of Łukasiewicz and Gödel logics, and provides empirical evidence of the good performance of SMT technology for automated theorem proving on infinitely-valued logics.

1 Introduction

There is a relatively large number of papers dealing with complexity and proof theory issues of infinitely-valued logics (see for example [1, 6, 7, 9] and the references therein). Nevertheless, little attention has been paid so far to the development of efficient solvers for such logics. To the best of our knowledge, most efforts have been devoted to Łukasiewicz infinitely-valued logic [4, 11]. It is also worth mentioning the translation from tableaux into mixed integer programming for certain infinitely-valued logics proposed in [8]. All such efforts are very limited compared with the work done for Boolean propositional logic. Nowadays, Boolean satisfiability solving (SAT) and its extensions (PBO, QBF, SMT, MaxSAT, ...) are recognized and active research lines with practical applications in relevant domains such as software and hardware verification, bioinformatics and planning.

In finitely-valued logics, signed CNF formulas provide a suitable and effective approach to satisfiability solving. Remind that any finitely-valued formula is polynomially reducible to a satisfiability preserving signed CNF formula, which can be tested with either a many-valued SAT solver or a Boolean SAT solver [3]. Unfortunately, there does not exist a clausal form transformation for dealing with any infinitely-valued logic. Therefore, it is not clear how SAT solvers can be used in the infinitely-valued case.

In this paper we propose to use Satisfiability Modulo Theories (SMT) to solve the satisfiability of infinitely-valued logics, taking Łukasiewicz, Gödel and Product logics as a case study. As we will see, SMT provides natural and compact encodings of infinitely-valued logics, and avoids the need of having a clausal form transformation. On the other hand, the mature state of SMT technology leads to very efficient theorem provers, and it would be very costly to duplicate the same efforts for developing efficient theorem provers based on a tableaux calculus [11] or a sequent calculus [4].

We also report on an empirical investigation to evaluate the performance of our approach, and compare the practical complexity of Łukasiewicz and Gödel logics. To this end, we have created a test suite which is useful for any t-norm logic. It consists of a number of theorems of Basic Logic [9], with tunable hardness, which are easy to generate.

This paper is structured as follows. Section 2 defines the notion of many-valued propositional logic, and introduces Łukasiewicz, Gödel and Product logics. Section 3 presents a brief introduction to SMT. Section 4 describes SMT-based theorem provers for Łukasiewicz and Product logics. Section 5 presents the test suite we have implemented for t-norm logics. Section 6 reports on the empirical investigation. Section 7 presents the conclusions and future work.

[*]Research partially supported by the Generalitat de Catalunya under grant AGAUR 2009-SGR-1434, the *Ministerio de Economía y Competividad* research projects AT CONSOLIDER CSD2007-0022, INGENIO 2010, TIN2008-04547, TIN2009-14704-C03-01, and TIN2010-20967-C04-01/03, and Newmatica INNPACTO IPT-2011-1496-310000.

2 Infinitely-Valued Logics

We define basic concepts of many-valued propositional logics, and then introduce the syntax and semantics of Łukasiewicz, Gödel and Product logics.

Definition 1. A propositional language is a pair $\mathbb{L} =< \Theta, \alpha >$, 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 logical constants. If $\Theta = \{\theta_1, \ldots, \theta_r\}$, in the following we will denote the propositional language $\mathbb{L} =< \Theta, \alpha >$ with $< \theta_1/\alpha(\theta_1), \ldots, \theta_r/\alpha(\theta_r) >$.

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

Definition 2. If $\mathbb{L} =< \Theta, \alpha >$ is a propositional language, N is a truth value set and A assigns to each $\theta \in \Theta$ a function $A_\theta : N^{\alpha(\theta)} \to N$, then we call a pair $\mathbf{A} =< N, A >$ a matrix for $\mathbb{L}$.

A pair $\mathcal{L} =< \mathbb{L}, \mathbf{A} >$ consisting of a propositional language and a matrix is called a many-valued logic.

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 is a function $I : \Sigma \to N$. I is extended to arbitrary formulas ϕ in the usual way:

1. If ϕ is a logical constant, the $I(\phi) = A(\phi)$.
2. If $\phi = \theta(\varphi_1, \ldots, \phi_r)$, then $I(\theta(\varphi_1, \ldots, \phi_r)) = A_\theta(I(\varphi_1), \ldots, I(\phi_r))$.

A formula ϕ is satisfiable iff there is an interpretation such that $I(\phi) \in D$, and it is a tautology iff $I(\phi) \in D$ for every interpretation.

In the rest of the section we concentrate on t-norm based fuzzy propositional logics [9]. More specifically, we focus on Łukasiewicz logic (Ł), Gödel logic (G) and Product logic (Π), which are the most representative fuzzy logics and can be obtained from Basic Fuzzy Logic (BL) by adding one or two additional axioms. Moreover, each continuous t-norm is an ordinal sum of isomorphic copies of the t-norms of Ł, G and Π [10].

From now on, we assume that a formula ϕ of an infinitely-valued logic is satisfiable iff there is an interpretation such that $I(\phi) = 1$, and it is a tautology iff $I(\phi) = 1$ for every interpretation.

Definition 4. The language of Łukasiewicz logic is given by

$$\mathbb{L}_{Łuk} =< \neg/1, \to/2, \wedge/2, \vee/2, \odot/2, \oplus/2 > .$$

Subsets of the set of connectives such as $\{\to, \neg\}$ and $\{\oplus, \neg\}$ are functionally complete. Nevertheless, we consider the whole set of connectives to illustrate how they are encoded into SMT. We refer to $\neg$ as negation, refer to $\to$ as implication, refer to $\wedge$ as weak conjunction, refer to $\vee$ as weak disjunction, refer to $\odot$ as strong conjunction, and refer to $\oplus$ as strong disjunction.

Definition 5. The Łukasiewicz infinitely-valued logic is the many-valued logic such that N is the real unit interval $[0,1]$, $\mathbb{L} = \mathbb{L}_{Łuk}$, $D = \{1\}$, and the matrix is given by:

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

Definition 6. The language of Gödel logic is given by

$$\mathbb{L}_G =< \neg/1, \to/2, \wedge/2, \vee/2 > .$$

Definition 7. The Gödel infinitely-valued logic is the many-valued logic such that N is the real unit interval $[0,1]$, $\mathbb{L} = \mathbb{L}_G$, $D = \{1\}$, and the matrix is given by:

$$\begin{aligned} A_\neg(x) &= \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x > 0 \end{cases} \\ A_\to(x,y) &= \begin{cases} 1 & \text{if } x \le y \\ y & \text{if } x > y \end{cases} \\ A_\wedge(x,y) &= \min\{x,y\} \\ A_\vee(x,y) &= \max\{x,y\} \end{aligned}$$

In Gödel logic we do not make distinction between weak and strong conjunction because both are interpreted by the same truth functions.

Definition 8. The language of Product logic is given by

$$\mathbb{L}_\Pi =< \neg/1, \to/2, \wedge/2, \vee/2, \odot/2, \oplus/2 > .$$

Definition 9. The Product infinitely-valued logic is the many-valued logic such that N is the real unit interval $[0,1]$, $\mathbb{L} = \mathbb{L}_\Pi$, $D = \{1\}$, and the matrix is given by:

$$\begin{aligned} A_\neg(x) &= \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x > 0 \end{cases} \\ A_\to(x,y) &= \begin{cases} 1 & \text{if } x \le y \\ y/x & \text{if } x > y \end{cases} \\ A_\wedge(x,y) &= \min\{x,y\} \\ A_\vee(x,y) &= \max\{x,y\} \\ A_\odot(x,y) &= x \cdot y \\ A_\oplus(x,y) &= x + y - x \cdot y \end{aligned}$$

3 Satisfiability Modulo Theories (SMT)

An SMT instance is a generalization of a Boolean formula in which some propositional variables have been replaced by predicates with predefined interpretations from background theories. For example, a formula can contain clauses like, e.g., $p \vee q \vee (x + 2 \leq y) \vee (x > y + z)$, where p and q are Boolean variables, and x, y and z are real variables. Predicates over non-Boolean variables, such as linear real inequalities, are evaluated according to the rules of a background theory. Examples of theories include linear real or integer arithmetic, arrays, bit vectors, etc., or combinations of them.

Formally speaking, a *theory* is a set of first-order formulas closed under logical consequence. The SMT problem for a theory T is: given a first-order formula F, determine whether there is a model of $T \cup \{F\}$.

Although an SMT instance can be solved by encoding it into an equisatisfiable SAT instance and feeding it to a SAT solver, currently most successful SMT solvers are based on the integration of a SAT solver and a T-solver, that is, a decision procedure for the given theory T. In this so-called lazy approach, while the SAT solver is in charge of the Boolean component of reasoning, the T-solver deals with sets of atomic constraints in T. The main idea is that the T-solver analyzes the partial model that the SAT solver is building, and warns it about conflicts with theory T (T-inconsistency). This way, we are hopefully getting the best of both worlds: in particular, the efficiency of the SAT solver for the Boolean reasoning and the efficiency of special-purpose algorithms inside the T-solver for the theory reasoning. See [13] for a survey on this approach.

Among the theories considered in the SMT library we are interested in the following logics [2]:

- QF_LRA: quantifier-free *linear real arithmetic*. In essence, closed quantifier-free formulas with Boolean combinations of inequalities between linear polynomials over real variables.
- QF_NRA: quantifier-free *non-linear real arithmetic*. It is like QF_LRA logic but allowing non-linear expressions.

4 Automated Theorem Provers for Infinitely-Valued Logic

The aim of this section is to show that SMT technology can indeed be used to build automated theorem provers for infinitely-valued logics, taking Łukasiewicz and Product infinitely-valued logics as a case study. As we will see, such logics can be encoded into SMT in a natural and compact way. We do not show Gödel logic due to the lack of space, but its formulation is similar.

4.1 The SMT-based Theorem Prover for Łukasiewicz Infinitely-Valued Logic

We describe a theorem prover capable of determining whether a formula ϕ of the Łukasiewicz infinitely-valued logic is a tautology. To this end, we develop a satisfiability checker for Łukasiewicz logic, and we ask whether there is an interpretation I such that $I(\phi) < 1$. If such an interpretation does not exist, then ϕ is a tautology.

Figure 1 shows the code of the SMT-based theorem prover for Łukasiewicz logic in the SMT-LIB language v2.0 under QF_LRA [2]. We find three sections. The first one determines the logic to be used: quantified free linear real arithmetic (QF_LRA). The second one declares the connectives of Łukasiewicz logic, and the variables occurring in the formula. Notice that the variable domains are restricted to the real unit interval. The `ite` (if-then-else) SMT function `(ite Bool s s)` returns its second argument or its third depending on whether its first argument is true or not.

The third one contains the following query: Is there an interpretation I of the formula $\phi = (x \oplus x) \vee (y \oplus y) \rightarrow (x \vee y) \oplus (x \vee y)$ such that $I(\phi) < 1$. ϕ is a tautology because the theorem prover returns unsatisfiable. If we would like to determine whether another formula is a tautology, we should only replace the formula and leave the rest of the program unchanged.

4.2 The SMT-based Theorem Prover for Product Infinitely-Valued Logic

Figure 2 shows the code of the SMT-based theorem prover for Product logic in the SMT-LIB language v2.0 under QF_NRA [2]. We also find three sections. The first one determines the logic to be used: quantified free non-linear real arithmetic (QF_NRA). The second one declares the connectives. Observe that now the defined functions correspond to the truth functions of Product logic. The third one contains the query, which corresponds to axiom A2 of BL for $n = 1$, defined in the next section.

5 A test Suite for t-Norm Logics

Basic Logic (BL) offers a framework for defining the most representative fuzzy logics by adding additional axioms to the axioms of BL. This means that the axioms of BL defined in [9] are valid in all the t-norm based fuzzy logics, including Ł, G and Π. Our test suite is formed by instances of the first six axioms of BL defined as follows:

- (A1) $(\varphi^n \rightarrow \psi^n) \rightarrow ((\psi^n \rightarrow \chi^n) \rightarrow (\varphi^n \rightarrow \chi^n))$
- (A2) $(\varphi^n \odot \psi^n) \rightarrow \varphi^n$
- (A3) $(\varphi^n \odot \psi^n) \rightarrow (\psi^n \odot \varphi^n)$

```
(set-logic QF_LRA)

; min(x,y)
(define-fun min ((x Real) (y Real)) Real
 (ite (> x y) y x))
; max(x,y)
(define-fun max ((x Real) (y Real)) Real
 (ite (> x y) x y))
; strong disjunction: sdis(x,y) = min{1,x+y}
(define-fun sdis ((x Real) (y Real)) Real
 (min 1 (+ x y)))
; strong conjuction: scon(x,y) = max{0,x+y-1}
(define-fun scon ((x Real) (y Real)) Real
 (max 0 (- (+ x y) 1)))
; weak disjunction: wdis(x,y) = max{x,y}
(define-fun wdis ((x Real) (y Real)) Real
 (max x y))
; weak conjunction: wcon(x,y) = min{x,y}
(define-fun wcon ((x Real) (y Real)) Real
 (min y x))
; negation: neg(x) = 1 - x
(define-fun neg ((x Real)) Real
 (- 1 x))
; implication: impl(x,y) = min{1,1-x+y}
(define-fun impl ((x Real) (y Real)) Real
 (min 1 (- (+ 1 y) x)))
(declare-fun x () Real)
(declare-fun y () Real)
(assert (>= x 0))
(assert (<= x 1))
(assert (>= y 0))
(assert (<= y 1))

(assert (< (impl (wdis (scon x x) (scon y y))
               (scon (wdis x y) (wdis x y))) 1))
(check-sat)
```

Figure 1. Code of the SMT-based theorem prover for Łukasiewicz infinitely-valued logic.

```
(set-logic QF_NRA)

; min(x,y)
(define-fun min ((x Real) (y Real)) Real
 (ite (> x y) y x))
; max(x,y)
(define-fun max ((x Real) (y Real)) Real
 (ite (> x y) x y))
; strong disjunction: sdis(x,y) = x+y - x*y
(define-fun sdis ((x Real) (y Real)) Real
 (- (+ x y) (* x y)))
; strong conjuction: scon(x,y) = x*y
(define-fun scon ((x Real) (y Real)) Real
 (* x y))
; weak disjunction: wdis(x,y) = max{x,y}
(define-fun wdis ((x Real) (y Real)) Real
 (max x y))
; weak conjunction: wcon(x,y) = min{x,y}
(define-fun wcon ((x Real) (y Real)) Real
 (min y x))
; negation: neg(x) = 1 if x=0; 0 if x>0
; we assume x in [0,1]
(define-fun neg ((x Real)) Real
 (ite (= x 0) 1 0))
; implication: impl(x,y) = y/x if x>y; 1 otherwise
(define-fun impl ((x Real) (y Real)) Real
 (ite (> x y) (/ y x) 1))
(declare-fun x () Real)
(declare-fun y () Real)
(declare-fun x1 () Real)
(assert (>= x 0) )
(assert (<= x 1) )
(assert (>= y 0) )
(assert (<= y 1) )

(assert (< (impl (scon x y) x) 1))
(check-sat)
```

Figure 2. Code of the SMT-based theorem prover for Product infinitely-valued logic.

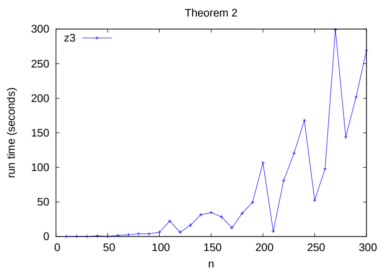

Figure 3. Łukasiewicz logic: Axiom A2

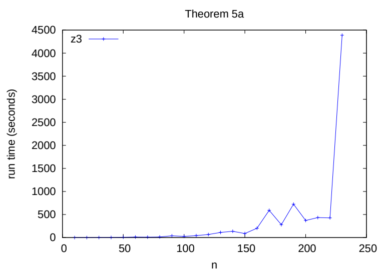

Figure 4. Łukasiewicz logic: Axiom A5a

- (A4) $(\varphi^n \odot (\varphi^n \to \psi^n)) \to (\psi^n \odot (\psi^n \to \varphi^n))$
- (A5a) $(\varphi^n \to (\psi^n \to \chi^n)) \to ((\varphi^n \odot \psi^n) \to \chi^n)$
- (A5b) $((\varphi^n \odot \psi^n) \to \chi^n) \to (\varphi^n \to (\psi^n \to \chi^n))$
- (A6) $((\varphi^n \to \psi^n) \to \chi^n) \to (((\psi^n \to \varphi^n) \to \chi^n) \to \chi^n)$

where $\phi^1 = p$ and $\phi^n = p \odot \phi_i^{n-1}$ for $\phi \in \{\varphi, \psi, \chi\}$; and p is a propositional variable.

6 Empirical Investigation

We performed an empirical evaluation of our SMT-based theorem provers. We ran experiments on a machine with 2.5 GHz and 0.5G mem. The SMT solver used in our experimentation is Z3 [5] (version 3.2) with the QF_LRA logic. We do not show results for Product Logic because the implementation of QF_NRA in Z3 is limited[1].

Figure 3 shows the results for axiom A2 of BL, ranging from $n = 1$ to $n = 300$ with increments of 10, with the described SMT-based theorem prover for Łukasiewicz infinitely-valued logic. Figure 4 is like Figure 3 but for axiom A5a of BL, and Figure 5 is like Figure 3 but for axiom A6 of BL.

[1]In fact, Z3 does not support expressions like x/y where both x and y are variables. We have made our experiments by indirectly encoding division with multiplication and auxiliary variables, but still Z3 has just been able to solve a few small instances.

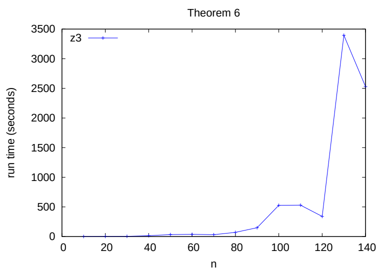

Figure 5. Łukasiewicz logic: Axiom A6

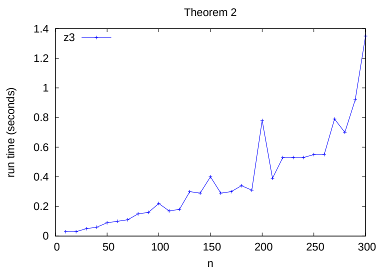

Figure 6. Product logic: Axiom A2

Figure 6 shows the results for axiom A2 of BL, ranging from $n = 1$ to $n = 300$ with increments of 10, with the described SMT-based theorem prover for Gödel infinitely-valued logic. Figure 7 is like Figure 6 but for axiom A5a of BL, and Figure 8 is like Figure 6 but for axiom A6 of BL.

In all the figures, the x-axis represents the value of n and the y-axis represents the time, in seconds, needed to prove the theorem. Due to the lack of space, we do not display results for the other axioms of BL, but we have obtained similar results.

Observe that the difficulty of the instances increases with n for both Łukasiewicz and Gödel logics. So, we can adjust the hardness of the benchmarks. This is a desired property for comparing the scaling behavior of different theorem provers. Also notice that the benchmark theorems hold for any fuzzy logic, not only for Łukasiewicz and Gödel logics.

On the other hand, observe that Łukasiewicz theorems are harder to prove in practice, despite that the tautology problem is NP-complete in both logics. This is due to the overhead of the arithmetic operations in the truth functions of Łukasiewicz logic. We claim that the results for Product logic will be even harder because the overhead of non-linear operations is bigger.

Notice that there are some peaks in the plots. The increase is not completely monotone because of the heuristics implemented in SMT solvers.

It is worth commenting that the theorems are proved in a reasonable amount of time taking into account the length of the instances. This performance profile is difficult to

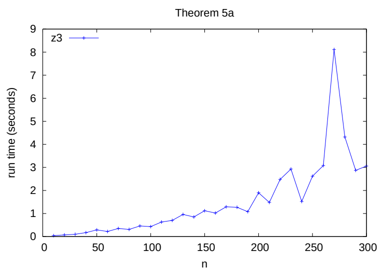

Figure 7. Product logic: Axiom A5a

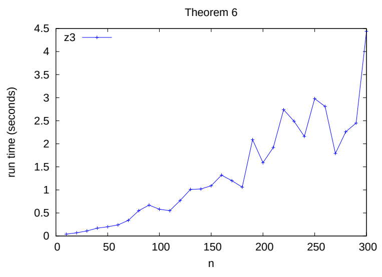

Figure 8. Product logic: Axiom A6

reach using a Prolog implementation as it was done for Łukasiewicz logic in [12].

7 Conclusions

We have presented how SMT technology can be used to define theorem provers for fuzzy logics, built a test suite for those logics, and performed a preliminary empirical evaluation. We would like to highlight that the code of the described theorem provers provides a natural and compact representation of the logics. Moreover, the code can be understood without needing a deep knowledge on SMT, and the described theorem provers can easily be adapted to other fuzzy logics.

Our work can be seen as a position paper in the sense that opens a new research line, consisting in exploiting SMT technology in fuzzy logics. To the best of our knowledge there has not been established before a link between SMT and many-valued logics.

There are a number of tasks and open questions that we propose as future work:

- Perform an empirical comparison between described theorem provers for infinitely-valued logics and our SMT-based approach. In particular, it would be interesting to compare with the approach described in [8].
- Perform experiments for Product logic using an SMT-based theorem prover with an efficient implementation of non-linear real arithmetic.
- Create a webpage containing a repository with benchmarks and theorem provers for fuzzy logics.
- Solve real applications with SMT-based theorem provers. The non-existence of fast and modern theorems provers has limited so far the potential of fuzzy logics in some real-world applications.
- Stimulate the development of efficient theorem provers and the creation of challenging benchmarks by organizing a competition.
- Evaluate the performance profile of SMT technology for finitely-valued logics.

References

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

[2] C. Barrett, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `http://www.SMT-LIB.org`, 2010.

[3] B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.

[4] A. Ciabattoni, C. G. Fermüller, and G. Metcalfe. Uniform rules and dialogue games for fuzzy logics. In F. Baader and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings*, volume 3452 of *LNCS*, pages 496–510. Springer, 2005.

[5] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340, 2008.

[6] S. Gottwald. *A Treatise on Many-Valued Logics*. Research Studies Press, Baldock, Hertfordshire, England, 2001.

[7] S. Gottwald. Mathematical fuzzy logic. *The Bulletin of Symbolic Logic*, 14(2):210–239, 2008.

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

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

[10] P. S. Mostert and A. L. Shields. On the structure of semigroups on a compact manifold with boundary. *Annals of Mathematics*, 65:117–143, 1957.

[11] N. Olivetti. Tableaux for Łukasiewicz infinitely-valued logic. *Studia Logica*, 73(1):81–111, 2003.

[12] R. Rothenberg. A class of theorems in Łukasiewicz logic for benchmarking automated theorem provers. In *16th International Conference, TABLEAUX 2007, Aix en Provence,France*, 2007. Position paper.

[13] R. Sebastiani. Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation*, 3(3-4):141–224, 2007.