Automated Proofs in Lukasiewicz Logic

Siva Anantharaman
LIFO, Dépt. Math-Info.
Université d'Orléans
45067 Orléans Cedex 02
FRANCE
E-mail: siva@univ-orleans.fr

Maria Paola Bonacina*
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400 USA
E-mail: bonacina@sbcs.sunysb.edu

November 28, 1989

Abstract

In this paper we present some mechanical proofs in the many-valued logic defined by Lukasiewicz. The main result is the first mechanical proof of the *fifth Lukasiewicz conjecture*. All the presented proofs are obtained by the (AC-)Unfailing Knuth-Bendix completion method in the theorem prover SBR3. However, a proof of the conjecture cannot be obtained by a brute-force application of completion. We introduce auxiliary functions and reformulate the theorem in terms of the new auxiliary functions. No manual addition of lemmas is needed. These transformations are customary in mathematics and we feel they are useful in automated reasoning too.
The paper is organized as follows: in section 1 we present the fifth Lukasiewicz conjecture, in section 2 we briefly present the prover SBR3 and in section 3 we present the proofs.

1 A problem in many-valued logic

Many-valued propositional logic was first introduced by Jan Lukasiewicz in the 1920's. All the following results about early work on many-valued logic are reported in [8].

The original definition of many-valued logic is purely semantical. No axioms and no inference rules are given. Lukasiewicz defines first a model and then the logic is defined as the set of all sentences in propositional calculus which are true on that model. More precisely, the n-valued logic L_n is defined as the set of all sentences satisfied by the structure

$$\mathcal{L}_n =< \{\frac{k}{n-1} | 0 \leq k \leq n-1\}, g, f >$$

where $A_n = \{\frac{k}{n-1} | 0 \leq k \leq n-1\}$ is the domain, $g : A_n \rightarrow A_n$ is the unary function $g(x) = 1 - x$ and $f : A_n \times A_n \rightarrow A_n$ is the binary function $f(x, y) = min(1 - x + y, 1)$.

L_1 is the set of all legal propositional sentences, L_2 is classical two-valued propositional logic with model

$$\mathcal{L}_2 =< \{0, 1\}, g, f >$$

*Research supported in part by grants CCR-8805734 & INT-8715231, both funded by the National Science Foundation, and by Dottorato di ricerca in Informatica, Universitá degli Studi di Milano.

where the functions g and f are classical negation and implication. L_3 is three-valued logic, the first one introduced by Lukasiewicz. As n increases, the domain A_n grows:

$A_1 = \{1\}$

$A_2 = \{0, 1\}$

$A_3 = \{0, \frac{1}{2}, 1\}$

$A_4 = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$

$A_5 = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\} \ldots$

The limit of this sequence is the set Q_0 of all rational numbers in the interval $[0, 1]$, which is the domain of the many valued logic

$$\mathcal{L}_{\aleph_0} =< \{\frac{k}{l} | 0 \leq k \leq l\}, g, f >$$

As n increases, the set L_n shrinks and $L_{\aleph_0}$ is the smallest such set, i.e. the intersection of all the L_n. It has been later proved by Lindenbaum that the domain for $\mathcal{L}_{\aleph_0}$ can be any arbitrary set of numbers $\{x | 0 \leq x \leq 1\}$ closed with respect to g and f.

The following research in the 20's and 30's was devoted to find axiomatizations for these n-valued logics. Wajsberg gave first an axiomatization for L_3 and proved that for all n such that $n - 1$ is a prime number, an axiomatization for L_n exists. Lindenbaum later extended this result to all n. Lukasiewicz conjectured that the following axioms are an axiomatization for $L_{\aleph_0}$:

1. $p \Rightarrow (q \Rightarrow p)$
2. $(p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r))$
3. $((p \Rightarrow q) \Rightarrow q) \Rightarrow ((q \Rightarrow p) \Rightarrow p)$
4. $((p \Rightarrow q) \Rightarrow (q \Rightarrow p)) \Rightarrow (q \Rightarrow p)$
5. $(not(p) \Rightarrow not(q)) \Rightarrow (q \Rightarrow p)$

where not and $\Rightarrow$ are interpreted as g and f in the model $\mathcal{L}_{\aleph_0}$. In the following we use not, $\Rightarrow$ and $true$ rather than g, f and 1 whenever we are working on the axiomatization rather than on the model. However, it should be clear that they are not negation, implication and truth in classical logic, but they are interpreted as g, f and 1 in $\mathcal{L}_{\aleph_0}$. The fourth axiom was later proved to be dependent on the remaining ones.

In the following years classes of algebras related to Lukasiewicz logics have been given an equational axiomatization. The following is the equational axiomatization for *Wajsberg algebras* [7]:

1. $true \Rightarrow x == x$
2. $(x \Rightarrow y) \Rightarrow ((y \Rightarrow z) \Rightarrow (x \Rightarrow z)) == true$
3. $(x \Rightarrow y) \Rightarrow y == (y \Rightarrow x) \Rightarrow x$
4. $(not(x) \Rightarrow not(y)) \Rightarrow (y \Rightarrow x) == true$

Axioms 2, 3 and 4 correspond to axioms 2, 3 and 5 in the original axiomatization by Lukasiewicz. We denote by $\mathcal{W}$ this equational presentation of Wajsberg algebras.

The problem we are interested in is the following: prove from $\mathcal{W}$ that

$$(x \Rightarrow y) \vee (y \Rightarrow x) == true$$

where the operator $\vee$ is defined by:

$$x \vee y == (x \Rightarrow y) \Rightarrow y.$$

This problem was first given by Lukasiewicz as a conjecture [9] and proved several years later [9]. It is known as the *fifth Lukasiewicz conjecture*. If the operator $\Rightarrow$ in the definition of the operator $\vee$ is interpreted as $min(1-x+y,1)$, then $x \vee y$ gets interpreted as $max(x,y)$:

$$min(1-min(1-x+y,1)+y,1) = \begin{cases} min(1-1+y,1) = y \ if \ y \geq x \\ min(1-1+x-y+y,1) = x \ if \ x \geq y \end{cases}$$

It follows that the theorem $(x \Rightarrow y) \vee (y \Rightarrow x) == 1$ is interpreted as $max(min(1-x+y,1), min(1-y+x,1)) == 1$, which is intuitively true, since the above evaluates to $max(1-x+y,1) = 1$ if $x \geq y$ and to $max(1-y+x,1) = 1$ if $y \geq x$. We are interested in obtaining a mechanical derivation of this theorem from the above axioms.

A major step to obtain the mechanical proof is the introduction of an auxiliary operator *or*, defined as

$$or(x,y) == not(x) \Rightarrow y$$

In the model $\mathcal{L}_{\aleph_0}$, *or* is interpreted as $min(1-1+x+y,1) = min(x+y,1)$. In Lukasiewicz logic $\vee$ and *or* are two different connectives, with two different interpretations. In order to distinguish them, we call them *weak disjunction* and *strong disjunction* respectively. In classical two-valued logic, i.e. on the domain $\{0,1\}$, the weak disjunction $x \vee y$ and the strong disjunction $or(x,y)$ are the same connective, the classical disjunction.

Finally, it is known that the following lemmas are true in any Wajsberg algebra [7]:

[1] $x \Rightarrow x == true$

[2] if $x \Rightarrow y == y \Rightarrow x == true$ then $x == y$

[3] $x \Rightarrow true == true$

[4] $x \Rightarrow (y \Rightarrow x) == true$

[5] if $x \Rightarrow y == y \Rightarrow z == true$ then $x \Rightarrow z == true$

[6] $(x \Rightarrow y) \Rightarrow ((z \Rightarrow x) \Rightarrow (z \Rightarrow y)) == true$

[7] $x \Rightarrow (y \Rightarrow z) == y \Rightarrow (x \Rightarrow z)$

[8] $x \Rightarrow false == x \Rightarrow not(true) == not(x)$

[9] $not(not(x)) == x$

[10] $not(x) \Rightarrow not(y) == y \Rightarrow x$

Some of these lemmas are involved in our mechanical proof of the fifth conjecture. All of them except lemma 7 are mechanically derived from $\mathcal{W}$ by SBR3 in running time varying from 2 secs to 1 minute, on a SUN 3/260. Lemma 7 is proved as well, but its deduction is not immediate as shown in the final section.

2 An overview of SBR3

SBR3 is written in CLU, and runs both on Sun-3 and the Vax-11 series. It is an enhancement of SbReve2 [2]. Its basic mechanism is the term rewriting method based on the well-known Knuth-Bendix completion.

The main features in SBR3 are the following:

1. AC-Unfailing Completion [2]
2. Cancellation inference rules [5]
3. Order saturation strategy [1]
4. Critical pair criteria for refutational theorem proving [3]
5. Heuristic strategies to sort equations [3]

SBR3 is a refutational theorem prover for proving equational theorems. The inputs to the prover are a set of equations, the axioms, and an inequality, the skolemized negation of the intended theorem. The user should also give a precedence among the function symbols and a status for each function symbol, so that an ordering can be imposed on the terms. The status can be left-to-right, multiset or right-to-left. The user also indicates which operators are AC and selects a sorting strategy.

3 The mechanical proofs

The fifth conjecture cannot be proved by simply applying (AC)UKB completion to the axiomatization $\mathcal{W}$ and the negation of the conjecture. This is not surprising since the mathematical proof of the conjecture took several years and it is unlikely that the numerous sophisticated steps of that proof are reducible to simplification and overlap steps alone in a straightforward way. Therefore, we adopt a different approach, in which the application of the completion procedure is controlled as follows: we break the proof into smaller steps, we introduce auxiliary operators to state the theorem in a different format and we exploit the possibility given by SBR3 to leave an equation, which could be oriented into a rewrite rule, unoriented. For example, we keep $not(not(x)) == x$ as an equation. This controlled application of completion allows us to prove the conjecture in about 24 minutes. Since reducing a problem to subproblems and finding more useful ways to write axioms and theorems are typical techniques in mathematics, our experiments with Lukasiewicz logic show a successful interaction of automated reasoning and mathematical reasoning.

Our mechanical proof of Lukasiewicz's conjecture is done incrementally through 5 executions:

Part 1. Prove

$$lemma\ 9:\ not(not(x)) == x$$

with $\mathcal{W}$ as input. During the proof the following lemmas are also generated automatically:

$$lemma\ 1:\ x \Rightarrow x == true$$

$$lemma\ 3:\ x \Rightarrow true == true$$

$$lemma\ 4: \ x \Rightarrow (y \Rightarrow x) == true$$

$$lemma\ 8: \ x \Rightarrow not(true) == not(x)$$

Part 2. Prove

$$lemma\ 10: \ not(x) \Rightarrow not(y) == y \Rightarrow x$$

with $\mathcal{W}$ and lemmas 1, 3 and 9 as input.

Part 3. Introduce the auxiliary operator *and* and prove

$$lemma\ 7: \ x \Rightarrow (y \Rightarrow z) == y \Rightarrow (x \Rightarrow z)$$

from $\mathcal{W}$ and lemmas 1, 3, 4, 8, 9 and 10.

Part 4. Introduce the auxiliary operator *or* defined as $or(x, y) == not(x) \Rightarrow y$, and prove that it is associative and commutative from lemmas 7, 9 and 10. Note that lemma 9 implies that $or(not(x), y) == x \Rightarrow y$ is equivalent to $or(x, y) == not(x) \Rightarrow y$.

Part 5. Prove

$$(x \Rightarrow y) \vee (y \Rightarrow x) == true$$

from $\mathcal{W}$, lemmas 1, 3, 9, $or(not(x), y) == x \Rightarrow y$, where *or* is AC, and $x \vee y == (x \Rightarrow y) \Rightarrow y$.

Previous knowledge of the lemmas of the Wajsberg algebras helps in breaking the proofs into smaller steps. This allows us to obtain the earlier lemmas as partial results and give them to the prover as additional inputs to prove further results. However we emphasize that the idea of introducing the auxiliary *or* and *and* is new and the observation that the lemmas imply that *or* is AC plays a crucial role in achieving the final result. All the proofs inside each of these parts are fully mechanical and all the lemmas are generated by SBR3 from the given axioms by pure forward reasoning: whenever the negation of one of the above lemmas is given to the prover as a goal, the prover achieves the refutation by actually generating the lemma itself.

In the various extracts given below from the output files, all the simplification steps used in the generation of rules or inequalities are also indicated, whenever it seems necessary for comprehension. All the runtimes are given for a SUN 3/260.

3.1 Part 1: proof of lemmas 1, 3, 4, 8, 9 from $\mathcal{W}$

Instead of giving separate proofs for these assertions, we choose to show here that the completely automatic proof obtained by SBR3 for lemma 9 ($not(not(x)) == x$), generates also directly lemmas 1, 3, 4 and 8. For that we input the initial equation set $\mathcal{W}$ and the negation of lemma 9: $not(not(cx)) \neq cx$, where cx is a Skolem constant.

Here is the extract from the script file for the proof, obtained in 58 secs., using the recursive path ordering induced by the precedence: $\Rightarrow > not > true$ with left-to-right status for $\Rightarrow$. Actually, any status or no status works as well here.

```
..........
New Rule (or Ineqn.)::
[9] (not(x) => not(true)) => x -> true
From:
[6] (not(x) => not(y)) => (y => x) -> true
and:
```

```
[5] true => x -> x
.......
New Rule (or Ineqn.)::
[15] (y => true) => true -> y => y
From:
[4] (x => y) => y == (y => x) => x
and:
[5] true => x -> x
.......
New Rule (or Ineqn.)::
[44] y => ((y => z) => z) -> true
From:
[7] (x => y) => ((y => z) => (x => z)) -> true
and:
[5] true => x -> x

New Rule (or Ineqn.)::
[46] (x => true) => (z => (x => z)) -> true
From:
[7] (x => y) => ((y => z) => (x => z)) -> true
and:
[5] true => x -> x
.......
New Rule (or Ineqn.)::
[114] z => z -> true                        %%..Lemma 1 Proved here...
From:
[44] y => ((y => z) => z) -> true
and:
[5] true => x -> x
.......
New Rule (or Ineqn.)::
[129] y => true -> true                     %%..Lemma 3 Proved here...
From:
[44] y => ((y => z) => z) -> true
and:
[15] (y => true) => true -> y => y

The equation or Rule:
[46] (x => true) => (z => (x => z)) -> true
was reduced to:
[131] z => (x => z) -> true                 %%..Lemma 4 Proved here...
.......
New Rule (or Ineqn.)::
[141] ((x => x1) => z) => (x1 => z) -> true
From:
[131] z => (x => z) -> true
and:
[7] (x => y) => ((y => z) => (x => z)) -> true
......
```

```
New Rule (or Ineqn.)::
[153] not(true) => z -> true
From:
[141] ((x => x1) => z) => (x1 => z) -> true
and:
[9] (not(x) => not(true)) => x -> true
.......
New Rule (or Ineqn.)::
[160] not(y) => (y => x) -> true
From:
[141] ((x => x1) => z) => (x1 => z) -> true
and:
[6] (not(x) => not(y)) => (y => x) -> true
......
New Rule (or Ineqn.)::
[172] (y => not(true)) => not(true) -> y
From:
[153] not(true) => z -> true
and:
[4] (x => y) => y == (y => x) => x
......
New Rule (or Ineqn.)::
[190] not(y => not(true)) => y -> true
From:
[172] (y => not(true)) => not(true) -> y
and:
[160] not(y) => (y => x) -> true
........
New Rule (or Ineqn.)::
[200] y => (not(y) => not(true)) -> true
From:
[190] not(y => not(true)) => y -> true
and:
[6] (not(x) => not(y)) => (y => x) -> true

The equation or Rule:
[19] (y => (not(y) => not(true))) => (not(y) => not(true)) -> y
was reduced to:
[202] not(y) => not(true) -> y
.......
New Rule (or Ineqn.)::
[212] y => not(true) -> not(y)              %%..Lemma 8 Proved here...
From:
[202] not(y) => not(true) -> y
and:
[172] (y => not(true)) => not(true) -> y

The equation or Rule:
[172] (y => not(true)) => not(true) -> y
```

```
was reduced to:
[213] not(not(y)) == y
......
The equation or Rule:
[0] not(not(cx)) =/= cx -> 1
was reduced to:
[223] cx =/= cx -> 1
                  *** Proved ***
Knuth-Bendix runtime:
  Total: 58 seconds.
  Computed 345 critical pairs and ordered 98 equations into rules.
```

3.2 Part 2: proof of lemma 10 from $\mathcal{W}$ and lemmas 1, 3 and 9

The ordering used in this automatic proof is exactly the same as in part 1. Here is the extract from the script file.

```
User equations:
                    true => x == x
                    x => x == true
                    x => true == true
                    (x => y) => ((y => z) => (x => z)) == true
                    (x => y) => y == (y => x) => x
                    (not(x) => not(y)) => (y => x) == true
                    not(not(x)) == x
User Inequations:   (not(cx) => not(cy)) =/= (cy => cx) -> 1
.......
New Rule (or Ineqn.)::
[17] (x => not(y)) => (y => not(x)) -> true
From:
[12] (not(x) => not(y)) => (y => x) -> true
and:
[11] not(not(x)) -> x
........
New Rule (or Ineqn.)::
[25] (x => x1) => (not(x1) => not(x)) -> true
From:
[17] (x => not(y)) => (y => not(x)) -> true
and:
[11] not(not(x)) -> x
.......
Crit. Pair::
  ((y => x) => (not(x) => not(y))) => (not(x) => not(y))
                                          ==  true => (y => x)
From:
[5] (x => y) => y == (y => x) => x
and:
[12] (not(x) => not(y)) => (y => x) -> true

 Simplified with: [8] true => x -> x
```

```
 Simplified with: [25] (x => x1) => (not(x1) => not(x)) -> true
 Simplified with: [8] true => x -> x
New Rule (or Ineqn.)::
[50] y => x == not(x) => not(y)
From:
[5] (x => y) => y == (y => x) => x
and:
[12] (not(x) => not(y)) => (y => x) -> true

Crit. Pair.
    (not(not(cy)) => not(not(cx))) =/= (cy => cx) == 1
From:
[50] y => x == not(x) => not(y)
and:
[0] (not(cx) => not(cy)) =/= (cy => cx) -> 1

 Simplified with: [11] not(not(x)) -> x
 Simplified with: [11] not(not(x)) -> x
New Rule (or Ineqn.)::
[51] (cy => cx) =/= (cy => cx) -> 1
From:
[50] y => x == not(x) => not(y)
and:
[0] (not(cx) => not(cy)) =/= (cy => cx) -> 1
                              *** Proved ***
Knuth-Bendix runtime:
  Total: 11 seconds.
  Computed 157 critical pairs and ordered 24 equations into rules.
```

3.3 Part 3: proof of lemma 7 from $\mathcal{W}$ and lemmas 1, 3, 4, 8, 9, 10

This proof is obtained by reformulating the problem in terms of the auxiliary operator *and*, whose definition is only implicit in the following user input equation:

$$and(x, y) \Rightarrow z == (x \Rightarrow (y \Rightarrow z))$$

No other axiom involving *and* is necessary. We prove that *and* is commutative and lemma 7 is a consequence. To allow the reformulation in terms of *and*, we naturally give here the right-to-left status to $\Rightarrow$. The status given to *and* is irrelevant. The precedence is: $\Rightarrow > and > not > true$.

```
User equations:
                  true => x == x
                  x => x == true
                  x => true == true
                  (x => y) => ((y => z) => (x => z)) == true
                  (x => y) => y == (y => x) => x
                  x => (y => x) == true
                  x => not(true) == not(x)
                  not(x) => not(y) == y => x
```

```
                  not(not(x)) == x
                  AND(x, y) => z == x => (y => z)
User Inequations: AND(cx, cy) =/= AND(cy, cx) -> 1
.......
Crit. Pair.::
    x => true == AND(x, z) => z
From:
[18] x => (y => z) -> AND(x, y) => z
and:
[12] x => x -> true

 Simplified with: [13] x => true -> true
New Rule (or Ineqn.)::
[37] AND(x, z) => z -> true
From:
[18] x => (y => z) -> AND(x, y) => z
and:
[12] x => x -> true
.......
Crit. Pair.::
    x => not(y) == AND(x, y) => not(true)
From:
[18] x => (y => z) -> AND(x, y) => z
and:
[16] x => not(true) -> not(x)

 Simplified with: [16] x => not(true) -> not(x)
New Rule (or Ineqn.)::
[39] x => not(y) -> not(AND(x, y))
From:
[18] x => (y => z) -> AND(x, y) => z
and:
[16] x => not(true) -> not(x)
.......
 Simplified with: [39] x => not(y) -> not(AND(x, y))
 Simplified with: [14] not(not(x)) -> x
 Simplified with: [37] AND(x, z) => z -> true
 Simplified with: [39] x => not(y) -> not(AND(x, y))
The equation or Rule:
[8] not(x) => not(y) == y => x
was reduced to:
[45] y => x -> not(AND(not(x), y))
........
 Simplified with: [45] y => x -> not(AND(not(x), y))
 Simplified with: [14] not(not(x)) -> x
The equation or Rule:
[39] x => not(y) -> not(AND(x, y))
was reduced to:
[57] not(AND(y, x)) == not(AND(x, y))
```

```
......
New Rule (or Ineqn.)::
[119] AND(x, y) == AND(y, x)
From:
[57] not(AND(y, x)) == not(AND(x, y))
and:
[14] not(not(x)) -> x

New Rule (or Ineqn.)::
[120] AND(cy, cx) =/= AND(cy, cx) -> 1
From:
[119] AND(x, y) == AND(y, x)
and:
[0] AND(cx, cy) =/= AND(cy, cx) -> 1
                                        *** Proved ***
Knuth-Bendix runtime:
  Total: 17 seconds.
  Computed 144 critical pairs and ordered 51 equations into rules.
```

3.4 Part 4: the pseudo-disjunction *or* is AC

Here we prove the crucial result that *or* is AC. This observation allows us to use AC-UKB during the final proof of the theorem. The commutativity and associativity of *or* can be proved mechanically, but in order to save space we report here a simple proof by hand:

Commutativity:

$or(x, y) = (not(x) \Rightarrow y)$ (definition)

$= not(y) \Rightarrow not(not(x))$ (by lemma 10)

$= not(y) \Rightarrow x$ (by lemma 9)

$= or(y, x)$ (definition).

Associativity:

$or(or(x, y), z) = (not(or(x, y)) \Rightarrow z)$ (definition)

$= not(z) \Rightarrow not(not(or(x, y)))$ (by lemma 10)

$= not(z) \Rightarrow or(x, y)$ (by lemma 9)

$= not(z) \Rightarrow (not(x) \Rightarrow y)$ (definition)

$= not(x) \Rightarrow (not(z) \Rightarrow y)$ (by lemma 7)

$= (not(x) \Rightarrow or(z, y)) = or(x, or(z, y))$ (definition)

$= or(x, or(y, z))$ (by commutativity).

3.5 Part 5: the fifth Lukasiewicz conjecture

For this step, we input the initial set of equations:

$$true \Rightarrow x == x$$

$$x \Rightarrow x == true$$

$$x \Rightarrow true == true$$

$$(x \Rightarrow y) \Rightarrow ((y \Rightarrow z) \Rightarrow (x \Rightarrow z)) == true$$
$$not(not(x)) == x$$
$$((x \Rightarrow y) \Rightarrow y == (y \Rightarrow x) \Rightarrow x$$
$$or(not(x), y) == x \Rightarrow y$$
$$x \vee y == (x \Rightarrow y) \Rightarrow y$$

and the initial inequality:

$$(cx \Rightarrow cy) \vee (cy \Rightarrow cx) \neq true$$

The input includes in this order axiom 1, lemma 1, lemma 3, axiom 2, lemma 9, axiom 3, the definition of $\Rightarrow$ in terms of *or*, the definition of $\vee$ and the negation of the theorem, where *cx* and *cy* are Skolem constants. Axiom 4 is not included; it is needed only in part 4. The final refutation is obtained by order-saturation on the inequalities, as shown by the following extract of the script file (where 'weakor' stands for $\vee$). The only AC-operator is *or*.

```
.....
 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[9] true => x -> x
was reduced to:
[14] OR(not(true), x) == x

 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[10] x => x -> true
was reduced to:
[15] OR(not(x), x) -> true

 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[11] x => true -> true
was reduced to:
[16] OR(not(x), true) -> true

 Simplified with: [13] x => y -> OR(not(x), y)
 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[8] weakor(x, y) == (x => y) => y
was reduced to:
[18] weakor(x, y) -> OR(not(OR(not(x), y)), y)

 Simplified with: [13] x => y -> OR(not(x), y)
 Simplified with: [13] x => y -> OR(not(x), y)
 Simplified with: [13] x => y -> OR(not(x), y)
 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[6] (x => y) => y == (y => x) => x
was reduced to:
[19] OR(not(OR(not(x), y)), y) == OR(not(OR(not(y), x)), x)
```

```
 Simplified with: [13] x => y -> OR(not(x), y)
 Simplified with: [13] x => y -> OR(not(x), y)
The equation or Rule:
[0] weakor(cx => cy, cy => cx) =/= true -> 1
was reduced to:
[21] weakor(OR(cy, not(cx)), OR(cx, not(cy))) =/= true -> 1

 Simplified with: [18] weakor(x, y) -> OR(not(OR(not(x), y)), y)
The equation or Rule:
[21] weakor(OR(cy, not(cx)), OR(cx, not(cy))) =/= true -> 1
was reduced to:
[25] OR(cx, not(OR(cx, not(OR(cy, not(cx))), not(cy))), not(cy))
                =/= true  -> 1
......
New Rule (or Ineqn.)::
[28] OR(true, x) -> true
From:
[16] OR(not(x), true) -> true
and:
[12] not(not(x)) -> x
....
Superposing::
[19] OR(not(OR(not(x), y)), y) == OR(not(OR(not(y), x)), x)
.......
New Rule (or Ineqn.)::
[33] OR(not(OR(not(x), not(y))), not(x)) -> OR(not(OR(x, y)), y)
From:
[19] OR(not(OR(not(x), y)), y) == OR(not(OR(not(y), x)), x)
and:
[12] not(not(x)) -> x
.......
New Rule (or Ineqn.)::
[156] OR(not(OR(not(x), x1)), not(x)) == OR(not(OR(not(x1), x)), not(x1))
From:
[33] OR(not(OR(not(x), not(y))), not(x)) -> OR(not(OR(x, y)), y)
and:
[12] not(not(x)) -> x

Crit. Pair.::
    OR(cx, not(OR(cx, not(OR(cy, not(cx))))),
       not(OR(cy, not(OR(cx, not(OR(cy, not(cx)))))))) =/= true == 1
From:
[156] OR(not(OR(not(x), x1)), not(x)) ==
           OR(not(OR(not(x1), x)), not(x1))
and:
[25] OR(cx, not(OR(cx, not(OR(cy, not(cx))), not(cy))), not(cy)) =/=
                                                        true -> 1
 Simplified with:
```

```
        [19] OR(not(OR(not(x), y)), y) == OR(not(OR(not(y), x)), x)
 Simplified with:
        [19] OR(not(OR(not(x), y)), y) == OR(not(OR(not(y), x)), x)
 Simplified with: [15] OR(not(x), x) -> true
 Simplified with: [28] OR(true, x) -> true
 Simplified with: [28] OR(true, x) -> true
 Simplified with: [28] OR(true, x) -> true
New Rule (or Ineqn.)::
[157] true =/= true -> 1
From:
[156] OR(not(OR(not(x), x1)), not(x)) ==
                                    OR(not(OR(not(x1), x)), not(x1))
and:
[25] OR(cx, not(OR(cx, not(OR(cy, not(cx))), not(cy))), not(cy)) =/=
                                                          true -> 1
                            *** Proved ***
Knuth-Bendix runtime:
  Total: 22:30.
  Computed 359 critical pairs and ordered 27 equations into rules.
```

References

[1] **S. Anantharaman, J. Hsiang**, Automated Proofs of the Moufang Identities in Alternative Rings, J. of Automated Reasoning, To appear, 1989-90.

[2] **S. Anantharaman, J. Hsiang, J. Mzali**, SbReve2: A term Rewriting Laboratory with (AC-)Unfailing Completion, Proc. RTA-1989, LNCS no. 335, Springer-Verlag.

[3] **S. Anantharaman, A. Andrianarivelo**, Heuristical Criteria in Refutational Theorem Proving, Research Report, Université d'Orléans (Fr.), no. 89-9, 1989, (Submitted at the DISCO-90).

[4] **J. Hsiang, M. Rusinowitch**, On word problems in equational theories, Proc. of the 14th ICALP, Springer-Verlag LNCS, Vol 267, pp 54-71, 1987.

[5] **J. Hsiang, M. Rusinowitch, K. Sakai**, Complete set of inference rules for the cancellation laws, IJCAI 87, Milano, Italy, 1987.

[6] **G. Peterson, M.E. Stickel**, Complete sets of reductions for some equational theories, J. Ass. Comp. Mach., Vol 28 no. 2, pp 233-264, 1981.

[7] **J.M.Font, A.J.Rodriguez, A.Torrens**, Wajsberg algebras, Stochastica, Vol. 8, No. 1, 5–31, 1984

[8] **A.Tarski, J.Lukasiewicz**, Investigations into the sentential calculus, Chapter IV in A.Tarski, Logic, Semantics and Metamathematics, 38–56, Clarendon Press, Oxford, 1956

[9] **D.Mundici**, Personal communication