Automated Reasoning Algorithm for Linguistic Valued Łukasiewicz Propositional Logic

Jun Liu
School of Computing and Mathematics University of Ulster N. Ireland, UK j.liu@ulster.ac.uk

Luis Martínez López
Department of Computer Science University of Jaén E-23071 Jaén, Spain. martin@ujaen.es

Yang Xu
Dept. of Mathematics Southwest Jiaotong University, China xuyang@home.swjtu.edu.cn

Zhirui Lu
School of Computing and Mathematics, University of Ulster N. Ireland, UK Lu-z@ulster.ac.uk

Abstract

In this paper, firstly a new automated reasoning algorithm based on Boolean logic is proposed and its theorems of soundness and completeness on validating the unsatisfiability of logic formulae are given. Then this procedure is extended to a linguistic valued Łukasiewicz propositional logic L(X) with truth-value in Łukasiewicz linguistic valued algebras, and an α-automated reasoning algorithm with respect to certain linguistic value level α in L(X) is given. Its theorems of soundness and completeness associated with the α-unsatisfiability of the logical formulae in L(X) are also proved. This reflects the symbolic approach acts by direct reasoning on linguistic truth values.

1. Introduction

Since the introduction of resolution in 1965 by Robinson [1], resolution-based automated reasoning has been extensively studied in the context of finding natural and efficient proof systems to support a wide spectrum of computational tasks. In essence, all of these methods were carried out as follows: firstly the problem to prove the validity of a theorem in classical Boolean logic is transformed into validating the unsatisfiability of a logical formula variation from this theorem. Then a resolution algorithm is constructed to prove the unsatisfiability of this logical formula. Generally, the soundness and completeness theorem of the algorithm will be also given. This resolution method is of great importance on mechanical theorem proving in classical logic.

As the use of non-classical logics becomes increasingly important in computer science, artificial intelligence and logic programming, the development of efficient automated theorem proving based on non-classical logic is currently an active area of research. e.g., for fuzzy logic, among others, see [2]-[7] and for many-valued logic (MVL), among others, see [8]-[21].

In spite of verifying and extending the classical automated reasoning approach, most important proposals and results use only the so-called Kleene implication ($p\rightarrow q = -p\vee q$, $-$ is the negation). It implies that the formulae of their logic are syntactically equivalent to the formulae in classical logic. It is well known that several implication connectives, which formalize different nuances of many-valued logics, have been introduced. Moreover, in using MVL as a tool for the modelling of approximate reasoning, it is important and necessary to have a sort of control on the truth-value assumed by the conclusion, once the truth-values of the premises are given. Compared to classical deduction, the semantics in MVL is much more complex and need to be paid more attention.

Among the various MVLs, the Łukasiewicz logic $L[0, 1]$ with truth-values in [0, 1] is considered to be one of the most attractive MVLs. The language of $L_{\aleph}$ has two primitive logical connective $\{\rightarrow, '\}$, where "$\rightarrow$" is the Łukasiewicz implication given by $a\rightarrow b=\min\{1, 1-a+b\}$ ($a, b\in[0, 1]$) and $a'=1-a$ is the negation operation. One can easily check that the truth-value degrees of the formulae derivable in the Kleene's system and in the Łukasiewicz system not, in general, coincide. Especially the Łukasiewicz implication connective is more general and not reducible to the other classical connectives (e.g., $'$, $\wedge$ and $\vee$), unlike the Kleene implication. This irreducibility, though semantically justifiable, complicates the calculus. As a first step towards a variant automated approach, it is important to deal with the implication connectives. Also in the light of semantic consideration, the development of a relatively efficient calculus for the Łukasiewicz system seems desirable.

Residuated lattice structure is a very popular algebraic structure for inexact concepts as shown by Goguen in [22]. It was shown in [23] that Łukasiewicz algebras, Heyting algebras, Post algebras, Gaines algebras, and MV-algebras etc., all define a residuated lattice. There have been considerable efforts about fuzzy logic based on a residuated lattice, where

Proceedings of the 37th International Symposium on Multiple-Valued Logic (ISMVL'07)
0-7695-2831-7/07 $20.00 © 2007 IEEE

Pavelka [24] and [28] systematically discussed propositional and first-order calculi with values in an enriched residuated lattice respectively.

More importantly, Pavelka showed in [24] that the only natural way of formalizing fuzzy logic for truth values in the unit interval [0, 1] is by using the Łukasiewicz's implication operator or some isomorphic forms of it. There have been some important investigations in [17], [19-21], [29-31] from different ways on proving systems and automated reasoning containing a more general implication, especially the Łukasiewicz implication. This kind of study can provide a useful framework for the development of approximate reasoning. .

On the other hand, we all know that as human beings we are bound to express ourselves in a natural language that uses words. The meanings of words are inherently, imprecise, vague, and fuzzy, mostly can be very qualitative in nature. Therefore, it is necessary to investigate natural language based reasoning under uncertainty within the realm of AI.

A nice feature of linguistic variables is that their values are structured, which makes it possible to compute the representations of composed linguistic values from those of their composing parts. In order for linguistic variables to be useful tools of analysis, one ought to be able to manipulate them through various operations. One can directly symbolically manipulate the linguistic variables themselves. Based on using algebraic operations on the linguistic variables themselves, there is no need to manipulate any sort of membership functions at all.

Based on the symbolic approaches and the above ideas, we characterize the set of linguistic values by a Lukasiewicz algebraic structure, investigate the corresponding logic systems with linguistic truth values, and automated reasoning based on linguistic truth-valued logic system as well. A key idea behind is to directly manipulate the available linguistic information and knowledge, i.e., the symbolic approach acts by direct computation on linguistic truth values.

In this paper, a new automated reasoning method which differs from the resolution principle is proposed. Firstly, this method is applied to Boolean logic and then applied to Łukasiewicz propositional logic $L(X)$ with truth-value in a Łukasiewicz linguistic-valued implication algebras to prove the satisfiability of logical formulae at a certain linguistic truth-value level, while the corresponding theorems of soundness and completeness are also proved.

This paper is organized as follows: Section 2 recalls the unsatisfiability in Boolean logic and introduces a new automated reasoning algorithm based on Boolean logic, as well as its theorems of soundness and completeness. Section 3 firstly recalls the Łukasiewicz linguistic-valued propositional logic system L(X), then the automated reasoning algorithm based on Boolean logic is extended to an α-automated reasoning algorithm in L(X), its soundness and completeness are also proved. The conclusion is included in Section 4.

2. An Automated Reasoning Method Based on Boolean Logic

2.1. Boolean Satisfiability

Assume that

(1) $S=C_1\wedge...\wedge C_n$ is a conjunctive normal form without redundant term, denoted by $S=\{C_1,\ldots, C_n\}$, where C_i is a disjunction of literals, called a clause;

(2) The set of literals in C_i is H_i, and denote the cardinality of H_i as $|H_i|$ and set $|H_i|=h_i$, $i=1,\ldots, n$.

Lemma 2.1 S is unsatisfiable iff $\forall\, p_i\in H_i$ ($i=1,\ldots, n$), there exist complementary pairs in $\{p_1, \ldots, p_n\}$.

Lemma 2.2 If S is unsatisfiable, then there exists $C_i\in S$ and $C_j\in S\text{-}\{C_i\}$ such that $\forall\, p\in H_i$, there exists $q\in H_j$ such that $p=\text{-}q$. Here "-" means the negation in Boolean logic.

Lemma 2.3 $S=C_1\wedge C_2$ is unsatisfiable if and only if C_1 and C_2 are single-literal clauses and $C_1=\text{-}C_2$.

2.2. Automated Reasoning Algorithm Based on Boolean Logic

The automated reasoning algorithm which differs from the resolution algorithm to determine the satisfiability of S is given as follows:

Automated Reasoning Algorithm

Termination of the algorithm:

If $S=\varnothing$, then S is satisfiable and the algorithm is terminated.

If $|S|=1$, then S is satisfiable and the algorithm is terminated.

Step 0: select D_1 from S such that:

(1) Every literal of D_1 has a complement in $S\text{-}\{D_1\}$;

(2) The number of literals in D_1 is the least in all the clauses satisfying (1).

If there exists more than one such D_1, then select one from them. Obviously, the number of literals in D_1 is the least in all the clauses of S.

If D_1 doesn't exist, then S is satisfiable, the algorithm is terminated;

If D_1 exists, let $S_1=S-\{D_1\}$ and $D_1^*(p)=D_1(p)-\{p_1\}$. Then go to Step 1.

Step 1: for $p_1\in D_1$, search the subsequence clause $D_2(p_1)$ of p_1 such that

(i) $D_2(p_1)\in S\text{-}\{D_1\}$;

(ii) There exists p_{21} in $D_2(p_1)$ such that there exist some complementary pairs in $\{p_1, p_{21}\}$;

(iii) The number of literals in $D_2(p_1)$ is the least in all the clauses, which satisfies (ii). Then we have:

(1) If $D_2(p1)\text{-}\{p21\}=\varnothing$.

(i) If D_1-$\{p_1\}=\varnothing$, then S is unsatisfiable, the algorithm is terminated.

(ii) If D_1-$\{p_1\} \neq \varnothing$, then let D_1=: D_1-$\{p_1\}$, back to Step 1.

(2) If $D_2(p1)$-$\{p21\}\neq\varnothing$, then go to Step 2.

Step 2: for $p_2 \in D_2(p_1)$-$\{p_{21}\}$, search its subsequence clause $D_3(p_2)$ such that

(i) $D_3(p_2) \in$ S-$\{D_1, D_2(p_1)\}$;

(ii) There exists p_{32} in $D_3(p_2)$ satisfying that there exist some complementary pairs in $\{p_1, p_2, p_{32}\}$ (generally this case isn't unique in that there exists the complementary pair in $\{p_1, p_{32}\}$ or in $\{p_2, p_{32}\}$), it only needs to select one;

(iii) The number of literals in $D_3(p_2)$ is the least in all the clauses, which satisfies (ii).

Moreover, if $D_3(p_2)$ doesn't exist, then S is satisfiable, the algorithm can be terminated.

If $D_3(p_2)$ exists, then follow the step below:

Write $E(p_2)=\{x \mid$ there exists the complementary pair in $\{p_1, p_2, x\}, x \in D_3(p_2)\}$.

(a) If $D_3(p_2)$-$E(p_2)=\varnothing$, let $D_2(p_1)$-$\{p_{21}\}$=: $D_2(p_1)$-$\{p_{21}, p_2\}$, back to Step 1;

(b) If $D_3(p_2)$- $E(p_2) \neq \varnothing$, then go to Step 3.

Step 3: for $p_3 \in D_3(p_2)$-$E(p_2)$, search its subsequence clause $D_4(p_3)$ such that:

(i) $D_4(p_3) \in$ S-$\{D_1, D_2(p_1), D_3(p_2)\}$;

(ii) There exists p_{43} in $D_4(p_3)$ such that there exist some complementary pairs in $\{p_1, p_2, p_3, p_{43}\}$ (generally this case isn't unique in that there exist some complementary pairs in $\{p_1, p_{43}\}$ or in $\{p_2, p_{43}\}$, or in $\{p_3, p_{43}\}$), it only needs to select one among them;

(iii) In all the clauses which satisfies (ii), the number of literals in $D_4(p_3)$ is the least.

If $D_4(p_3)$ doesn't exist, then S is satisfiable, the algorithm can be terminated;

If $D_4(p_3)$ exists, then follow the steps below:

Write $E(p_3)=\{x \mid$ there exist some complementary pairs in $\{p_1, p_2, p_3, x\}, x \in D_4(p_3)\}$.

(a) If $D_4(p_3)$-$E(p_3)=\varnothing$, then let $D_3(p_2)$-$E(p_2)$=: $D_3(p_2)$-$(E(p_2) \bigcup \{p_3\})$, go back to Step 2.

(b) If $D_4(p_3)$- $E(p_3) \neq \varnothing$, then go to Step 4.

……

Step k: for $p_k \in D_k(p_{k-1})$-$E(p_{k-1})$, search its subsequence clause $D_{k+1}(p_k)$ (k=4,…, n-1) such that:

(i) $D_{k+1}(p_k) \in$ S-$\{D_1, D_2(p_1), D_3(p_2), \ldots, D_k(p_{k-1})\}$;

(ii) There exists p_{k+1k} in $D_{k+1}(p_k)$ such that there exist some complementary pairs in $\{p_1, p_2, p_3, \ldots, p_k, p_{k+1k}\}$ (generally this case isn't unique in that there exist some complementary pairs in $\{p_1, p_{k+1k}\}$ or in $\{p_2, p_{k+1k}\}$, …, or in $\{p_k, p_{k+1k}\}$), it only needs to select one of them;

(iii) The number of literals in $D_{k+1}(p_k)$ is the least in all the clauses, which satisfies (ii).

If $D_{k+1}(p_k)$ doesn't exist, then S is satisfiable, the algorithm can be terminated;

If $D_{k+1}(p_k)$ exists, then follow the steps below:

Write $E(p_k)=\{x \mid$ there exist some complementary pairs in $\{p_1, p_2, p_3, \ldots, p_k, x\}, x \in D_{k+1}(p_k)\}$.

(a) If $D_{k+1}(p_k)$-$E(p_k)=\varnothing$, then let $D_k(p_{k-1})$-$E(p_{k-1})$=: $D_k(p_{k-1})$-$(E(p_{k-1}) \bigcup \{p_k\})$, go back to Step k-1.

(b) If $D_{k+1}(p_k)$- $E(p_k) \neq \varnothing$, then go to Step k+1.

……

Until the above algorithms are all terminated.

Theorem 2.1 (Soundness and completeness) $S=C_1 \wedge C_2 \wedge \ldots \wedge C_n$ is unsatisfiable iff the automated reasoning algorithm above terminates at Step 1.

Due to the space restriction, the proof is skipped.

Example 2.1 Let a clause $S=\{C_1=\text{-}P \vee \text{-}Q \vee R, C_2=P \vee R, C_3=Q \vee R, C_4=\text{-}R\}$. We use the algorithm to this clause:

Step 0: select D_1.

$D_1=C_4=\text{-}R$, go to Step 1.

Step 1: $P_1 \in D_1$ (i.e., $P_1=\text{-}R$), seach $D_2(P_1)$:

$D_2(P_2)=P \vee R$, $P_{21}=R$.

Judge: $D_2(P_2)$-$\{P_{21}\}\neq\varnothing$, go to Step 2.

Step 2: $D_2(P_1)$-$\{P_{21}\}=P$, $P_2=P$,

$D_3(P_2)=C_1=\text{-}P \vee \text{-}Q \vee R$,

$E(P_2)=\{\text{-}P, R\}$,

$D_3(P_2)$-$E(P_2)\neq\varnothing$, go to Step 3.

Step 3: $P_3 \in D_3(P_2)$-$E(P_2)=\text{-}Q$, $P_3=\text{-}Q$:

$D_4(P_3)=C_3=Q \vee R$,

$E(P_3)=\{Q, R\}$,

$D_4(P_3)$-$E(P_3)=\varnothing$, go back to Step 2.

$D_3(P_2)$- $E(P_2)= D_3(P_2)$-$(E(P_2) \cup \{P_3\})=\varnothing$

Step 2^*: $D_2(P_1)$-$\{P_{21}\}=D_2(P_1)$-$\{P_{21}, P_2\}=\varnothing$, go to Step 1.

Step 1^*: D_1-$\{P_1\}=\varnothing$.

According to the algorithm, the clause S is unsatisfiable.

3. α-Automated Reasoning Method Based on L(X)

3.1. Łukasiewicz Propositional Logic L(X)

Definition 3.1 [24] A residuated lattice is a structure $\langle L, \otimes, \rightarrow \rangle$, where

(1) $\mathbf{L}=\langle L, \leq, \vee, \wedge, \mathrm{O}, \mathrm{I}\rangle$ is a bounded lattice with the least element O and the greatest element I.

(2) $\langle \otimes, \rightarrow \rangle$ is an adjoint couple on L, i.e.,

(a) $\otimes$ is istone (ordering preserving) on $L \times L$;

(b) $\rightarrow$ is antitone (order reversing) in the first and isotone in the second variable on $L \times L$;

(c) for all $x, y, z \in L$ hold the adjointness condition or Galois correspondence: $x \otimes y \leq z$ iff $x \leq y \rightarrow z$

(3) $\langle L, \otimes, \mathrm{I}\rangle$ is a commutative monoid.

The operation $\otimes$ is called multiplication and $\rightarrow$ is called residuation.

Example 3.1 (Łukasiewicz algebra on [0, 1]). If the operations on [0, 1] are defined respectively as follows: $x \vee y=\max(x, y)$, $x \wedge y=\min(x, y)$, $x \rightarrow y=\min(1, 1\text{-}x+y)$, $x \otimes y=\max(0, x+y\text{-}1)$, $x'=1\text{-}x$, then this algebra is a residuated lattice, denoted by $L[0, 1]$.

Let L be a finite chain, $L=\{a_i;\ 1 \leq i \leq n\}$ and $\mathrm{O}=a_1<\ldots<a_n=\mathrm{I}$. If $'$: $L \rightarrow L$ is defined by $(a_i)'=a_{n+1-i}$, and $\rightarrow$

and $\otimes$: $L\times L \to L$ are defined by $a_i \to a_j = a_{n+j-i}$ and $a_i \otimes a_j = a_{i+j-n}$ respectively ($i, j \in \{1,..., n\}$). Then (L, $\vee$, $\wedge$, ', $\to$, $\otimes$) is a residuated lattice, denoted by L_n.

In the following, $L[0, 1]$ and L_n are collectively called *Łukasiewicz algebra* and denoted as L.

Definition 3.2 Let X be the set of propositional variables, $T=L\cup\{', \otimes, \to\}$ be a type with $ar(')=1$, $ar(\otimes)= ar(\to) = 2$ and $ar(a)=0$ for every $a \in L$. The propositional algebra of the Łukasiewicz propositional calculus on the set of propositional variables is the free T algebra on X and is denoted by $L(X)$.

Proposition 3.2 $L(X)$ is the minimal set Y which satisfies the following conditions:

(1) $X \bigcup L \subseteq Y$,

(2) If $p, q \in Y$, then $p\otimes q$, $p', p \to q \in Y$.

Note that L and $L(X)$ are the algebras with the same type T, where $T = L \bigcup \{', \otimes, \to\}$.

Definition 3.3 A valuation of $L(X)$ is a propositional algebra homomorphism $\gamma: L(X)\to L$.

Definition 3.4 Let $p\in L(X)$, $\alpha \in L$. If $\gamma(p)\geq\alpha$ for every valuation γ of $L(X)$, we say that p is valid by truth-value level α. If there exists a valuation γ of $L(X)$ such that $\gamma(p)\geq\alpha$, then p is called α-satisfiable.

Definition 3.5 Let $p\in L(X)$, $\alpha \in L$. p is said to be always false by truth-value level α (α-false in short) if for any valuation γ, $\gamma(p) \leq \alpha$..

Beginning from the normal form is the natural and usual way to discuss the satisfiability of the formula in classical logic. Considering the great extension in connectives and truth-values in $L(X)$, especially the implication connectives in $L(X)$ are more general and not reducible to the other classical connectives unlike the Kleene implication. As a first step towards a variant resolution, it is important to deal with implication connectives and consider the generalized normal form.

Definition 3.6 An L-valued propositional logical formula f is called an extremely simple form, in short ESF, if an L-valued propositional logical formula f^* is obtained by deleting any constant or literal or implication term appearing in f is not equivalent to f.

Definition 3.7 An L-valued propositional logical formula f is called an indecomposable extremely simple form, in short IESF, if f is an ESF containing no connectives other than implication connectives.

Definition 3.8 An IESF is called an n-IESF, if there exist n implication connectives occurring in f.

Definition 3.9 All the constants, literals and IESF´s are called generalized literals. Here, the definition of literal is the same as that in classical logic.

Definition 3.10 An L-valued propositional logical formula G is called a generalized clause (phrase), if G is a formula of the form:

$G=g_1 \vee \ldots \vee g_i \vee \ldots \vee g_n (G=g_1 \wedge \ldots \wedge g_i \wedge \ldots \wedge g_n)$

where g_i (i=1,..., n) are generalized literals.

A conjunction (disjunction) of finite generalized clauses (phrases) is called a generalized conjunctive (conjunctive) normal form.

Now we consider some necessary lemmas. In the propositional logic $L(X)$, we assume:

(1) $S=C_1\wedge C_2\wedge...\wedge C_n$ is a generalized conjunctive normal form without redundant term, also written as $S=\{C_1, C_2, \ldots, C_n\}$;

(2) Denoted the generalized literal set of C_i as H_i, also $|H_i|=h_i$, i=1, 2,…, n.

Lemma 3.1 $S= C_1\wedge C_2\wedge...\wedge C_n \leq \alpha$ if and only if $\forall\, p_i \in H_i$, i=1, …, n, $p_1\wedge\ldots\wedge p_n \leq \alpha$.

Lemma 3.2 Let x, y, z be literals in $L(X)$, $\alpha\in L$, $\alpha<$I, and suppose that there exists $\beta\in L$ such that $\beta \wedge (\beta\to\beta') > \alpha$, then $x \wedge (y\to z) > \alpha$.

Corollary 3.1 Let x, y, t, and z be literals in $L(X)$, and there exists $\beta\in L$ and $\alpha<$I such that $\beta \wedge (\beta\to\beta') > \alpha\in L$, then $(x\to y) \wedge (t\to z) > \alpha$.

3.2. Linguistic-valued Lukasiewicz algebras

Here we characterize the set of linguistic values by a Lukasiewicz algebra. In general, the value of a linguistic variable can be a linguistic expression involving a set of linguistic values such as "*high*," "*middle*," and "*low*," modifiers such as "*very*," "*more or less*" (called *hedges* [25], [26]) and connectives (e.g., "*and*," "*or*"). Let us consider the domain of the linguistic variable "*truth*": domain (*truth*)={*true, false, very true, possibly true, very false*, ...}, which can be regarded as a partially ordered set whose elements are ordered by their meanings and also regarded as an algebraically generated set from the generators G={*true, false*} by means of a set of linguistic modifiers M={*very, possibly*, ...}. The linguistic modifiers are strictly related to the notion of vague concept. The generators G can be regarded as the prime term, different prime terms correspond to the different linguistic variables.

Consider a set of linguistic hedges, e.g. H^+={*very, more or plus*}, H={*approximately, possibly, little*}, where H^+ consists of hedges which strengthen the meanings of "*true*" and the hedges in H weaken it. Put $H=H^+\cup H$. H^+, H can be ordered by the degree of strengthening or weakening, e.g., one may assume that *very*>*more*, *little*>*approximately*. We say that $a\leq b$ iff $a(True)\leq b(True)$ in the natural language, where a and b are linguistic hedges.

Applying the hedges of H to the primary term "*true*" or "*false*" we obtain a partially ordered set or lattice. In this purpose, the first step is to choose the basic ingredients that are used in the symbolic manipulation. This means that the analyst has to choose a context-dependent linguistic terms set to describe vague or imprecise information. We suppose that T levels of linguistic variables may be used. For example, suppose that T=5, and S_T={s_0='*Poor*', s_1='*Low*', s_2='*Average*', s_3='*High*', s_4='*Good*'} (t=0,…, 4). Here it is supposed that S_T is a finite and totally ordered term set. Any label, s_i, represents a possible

value for a linguistic variable, and have the following characteristics [27]:

1) The set is ordered: $s_i \le s_j$ if $i \le j$.

2) There is the negation operator: $\mathrm{Neg}(s_i) = s_j$ such that $j = T\text{-}1\text{-}i$.

3) There is the maximization operator: $\mathrm{Max}(s_i, s_j) = s_i$ if $s_j \le s_i$.

4) There is the minimization operator: $\mathrm{Min}(s_i, s_j) = s_i$ if $s_i \le s_j$.

Moreover, one can define implication ($\rightarrow$) based on the L_n structure, i.e.,

5) $\rightarrow$: $s_i \rightarrow s_j = s_{n+j-i}$ $(i, j \in \{0, 1, ..., T\}$

The corresponding t-norm $\otimes$ and the t-conorm $\oplus$ can be also given as according to the L_n structure and properties respectively. This L_n is called a *linguistic-valued Lukasiewicz algebra*. In the following, $L(X)$ denotes the propositional algebra of the linguistic-valued Łukasiewicz propositional calculus.

3.3. α-Automated reasoning algorithm in L(X)

The automated reasoning algorithm to determine the α-satisfiability of S is given as follows, here $\alpha \in L_n$ is a linguistic truth-value:

α-Automated Reasoning Algorithm

Step 0: select D_1 from S, such that the number of literals in D_1 is the least in all the clauses of S. Then go to Step 1.

Step 1: for $p_1 \in D_1$, search the subsequence clause $D_2(p_1)$ of p_1 such that

(i) $D_2(p_1) \in S\text{-}\{D_1\}$;

(ii) There exists p_{21} in $D_2(p_1)$ satisfying $p_1 \wedge p_{21} \le \alpha$;

(iii) The number of literals in $D_2(p_1)$ is the least in all the clauses satisfying (ii).

If $D_2(p_1)$ satisfying the conditions above doesn't exist, select $D_2(p_1)$ as the clause (with the least number of literral) in $S\text{-}\{D_1\}$, then go to Step 2 and set $D_2(p_1)\text{-}\{p_{21}\} =: D_2(p_1)$.

If $D_2(p_1)$ satisfying the conditions above exists,

A. If $D_2(p_1)\text{-}\{p_{21}\} = \varnothing$.

(i) If $D_1\text{-}\{p_1\} = \varnothing$, then the algorithm terminates;

(ii) If $D_1\text{-}\{p_1\} \ne \varnothing$, then let $D_1 =: D_1\text{-}\{p_1\}$, back to Step 1.

B. If $D_2(p_1)\text{-}\{p_{21}\} \ne \varnothing$, then go to Step 2.

Step 2: for $p_2 \in D_2(p_1)\text{-}\{p_{21}\}$, search its subsequence clause $D_3(p_2)$, such that

(i) $D_3(p_2) \in S\text{-}\{D_1, D_2(p_1)\}$;

(ii) There exists p_{32} in $D_3(p_2)$ satisfying $p_1 \wedge p_2 \wedge p_{32} \le \alpha$;

(iii) The number of literals in $D_3(p_2)$ is the least in all the clauses satisfying (ii).

If these $D_3(p_2)$ doesn't exist, select $D_3(p_2)$ as the clause (with the least number of literals) in $S\text{-}\{D_1, D_2(p_1)\}$, then go to Step 3 and set $D_3(p_2)\text{-}E(p_2) =: D_3(p_2)$.

If $D_3(p_2)$ exists, then do as follows:

Write $E(p_2) = \{x \mid p_1 \wedge p_2 \wedge x \le \alpha,\ x \in D_3(p_2)\}$.

A. If $D_3(p_2)\text{-}E(p_2) = \varnothing$, let $D_2(p_1)\text{-}\{p_{21}\} =: D_2(p_1)\text{-}\{p_{21}, p_2\}$, back to Step 1;

B. If $D_3(p_2)\text{-}E(p_2) \ne \varnothing$, then go to Step 3.

Step 3: for $D_3(p_2)\text{-}E(p_2)$, search its subsequence clause $D_4(p_3)$ such that

(i) $D_4(p_3) \in S\text{-}\{D_1, D_2(p_1), D_3(p_2)\}$;

(ii) There exists p_{43} in $D_4(p_3)$ satisfying $p_1 \wedge p_2 \wedge p_3 \wedge p_{43} \le \alpha$;

(iii) The number of literals in $D_4(p_3)$ is the least in the clauses satisfying (ii).

If these $D_4(p_3)$ doesn't exist, select $D_4(p_3)$ as the clause (with the least number of literals) in $S\text{-}\{D_1, D_2(p_1), D_3(p_2)\}$, then go to Step 4 and set $D_4(p_3)\text{-}E(p_3) =: D_4(p_3)$. If these $D_4(p_3)$ exist, then do as follows:

Write $E(p_3) = \{x \mid$ there exist complementary pairs in $\{p_1, p_2, p_3, x\},\ x \in D_4(p_3)\}$.

A. If $D_4(p_3)\text{-}E(p_3) = \varnothing$, then let $D_3(p_2)\text{-}E(p_2) =: D_3(p_2)\text{-}(E(p_2) \cup \{p_3\})$, back to Step 2.

B. If $D_4(p_3)\text{-}E(p_3) \ne \varnothing$, then go to Step 4.

……

Step k: for $D_k(p_{k-1})\text{-}E(p_{k-1})$, search its subsequence clause $D_{k+1}(p_k)$ such that

(i) $D_{k+1}(p_k) \in S\text{-}\{D_1, D_2(p_1), ..., D_k(p_{k-1})\}$;

(ii) There exists p_{k+1k} in $D_{k+1}(p_k)$ satisfying $p_1 \wedge p_2 \wedge p_3 \wedge ... p_k p_{k+1k} \le \alpha$;

(iii) The number of literals in $D_{k+1}(p_k)$ is the least in the clauses satisfying (ii).

If this $D_{k+1}(p_k)$ doesn't exist, select $D_{k+1}(p_k)$ as the clause (with the least number of literals) in $S\text{-}\{D_1, D_2(p_1), D_3(p_2), \ldots, D_k(p_{k-1})\}$, then go to Step k+1 and set $D_{k+1}(p_k)\text{-}E(p_k) =: D_{k+1}(p_k)$. If this $D_{k+1}(p_k)$ does exist, then do as follows:

Write $E(p_k) = \{x \mid$ there exist complementary pairs in $\{p_1, p_2, p_3, x\},\ x \in D_{k+1}(p_k)\}$.

A. If $D_{k+1}(p_k)\text{-}E(p_k) = \varnothing$, then let $D_k(p_{k-1})\text{-}E(p_{k-1}) =: D_k(p_{k-1})\text{-}(E(p_{k-1}) \cup \{p_k\})$, back to Step k-1.

B. If $D_{k+1}(p_k)\text{-}E(p_k) \ne \varnothing$, then go to Step k+1.

……

Until the above algorithms are all terminated.

Theorem 3.1 (Soundness and Completeness) $S = C_1 \wedge C_2 \wedge ... \wedge C_n \le \alpha$ iff the α -automated reasoning algorithm of $L(X)$ above terminates at Step 1.

4. Conclusions

Focused on automated reasoning algorithm, two phases of results were obtained in this paper: proposed an automated reasoning algorithm to validate the unsatisfiability of Boolean logic formula, and proved that this algorithm is sound and complete; then the algorithm was extended to establish an α-automated reasoning algorithm to validate the α-unsatisfiability of logic formulae in Łukasiewicz linguistic-valued propositional logic $L(X)$ with truth value in the Łukasiewicz linguistic-valued algebras, and also proved that this algorithm is sound and complete.

All these results will provide a theoretical support for further investigating automated reasoning method on linguistic truth-valued logic system.

Acknowledgements

The work has been partially supported by the Research Project TIN2006-02121 and the National Natural Science Foundation of P.R. China (Grant No. 60474022).

The authors also would like to thank the anonymous referees for their valuable suggestions in improving this paper and on future research work.

References

[1] J.P. Robinson, "A machine-oriented logic based on the resolution principle", *J. of A.C.M.*, 12: 23-41, 1965.

[2] R.C.T. Lee, "Fuzzy logic and the resolution principle", *Journal of A.C.M.*, 19: 109-119, 1972.

[3] X.H. Liu and H. Xiao, "Operator fuzzy logic and fuzzy resolution", *Proc. of the 5th IEEE Inter. Symp. on Multiple-Valued Logic* (*ISMVL'85*), Kingstion, Canada, pages: 68-75, 1985.

[4] M. Mukaidono, "Fuzzy inference of resolution style", in: *Fuzzy Sets and Possibility Theory"*, R.R. Yager (Ed.), Pergamon Press, New York, pages: 224-231, 1982.

[5] D. Dubois and H. Prade, "Resolution principle in possibilistic logic", *International Journal of Approximate Reasoning* 4 (1): 1-21, 1990.

[6] T.J. Weigert, J.P. Tsai and X.H. Liu, "Fuzzy operator logic and fuzzy resolution", *Journal of Automated Reasoning*, 10(1): 59 – 78, 1993.

[7] C.S. Kim, D.S. Kim, and J.S. Park, "A new fuzzy resolution principle based on the antonym", *Fuzzy Sets and Systems*, 113 (2): 299-307, 2000.

[8] C.G. Morgan, "Resolution for many-valued logics", *Logique et Analyses*, 19 (74-76): 311-339, 1976.

[9] E. Orlowska, "Mechanical proof methods for Post logics", *Logique et Analyses*, 28(110): 173-192, 1985.

[10] P.H. Schmitt, "Computational aspects of three-valued logic", in*: Proc. of the 8th Inter. Conf. on Automated Deduction*, J.H. Siekmann (Ed.), Springer, LNCS, pages: 190-198, 1985.

[11] R. Hahnle, *Automated Deduction in Multiple-Valued L*ogics, Oxford University Press, 1993.

[12] H.A. Blair and V.S. Subrahamnian, "Paraconsistent logic programming", *Theoretical Computer Science*, 68:135-154, 1989.

[13] M. Kifer and V.S. Subrahamnian, "Theory of generalized annotated logic programming and its application", *J. of Logic Prog.* 12: 335-367, (1992).

[14] J.J. Lu and L.J. Henschen, "The completeness of gp-resolution for annotated logic", *Inform. Process. Lett.* 44: 135-140, 1992.

[15] M. Baaz, C.G. Fermuller, "Resolution for many valued logics". In *Proc. of Logic Programming and Automated Reasoning* (LPAR'92), Voronkv, A. (Ed.). Springer, LNAI 624, page 107-118, 1992.

[16] Z. Stachniak, *Resolution Proof Systems: An Algebraic Theory*, Kluwer Academic Publisher, Netherlands, 1996.

[17] S. Lehmke. "A resolution-based axiomatisation of `bold' propositional fuzzy logic". In: *Fuzzy Sets, Logics, and Reasoning about Knowledge* (Eds. by D. Dubois, E. P. Klement, and H. Prade), Kluwer Academic Publishers, Applied Logic, 1999.

[18] V. Sofronie-Stokkermans, "Chaining techniques for automated theorem proving in finitely-valued logics". In: *Proc. of the 2000 ISMVL*. Portland, Oregon, pages: 337-344, 2000.

[19] Y. Xu, D. Ruan, E.E. Kerre., J. Liu, "α-resolution principle based on lattice-valued propositional logic LP(X)". *Information Sciences*, 130: 195-223, 2000.

[20] Y. Xu, D. Ruan, E.E. Kerre., J. Liu, "α-Resolution principle based on first-order lattice-valued logic LF(X)", *Information Sciences*, 132: 221-239, 2001

[21] J. Liu, D. Ruan, Y. Xu, Z.M. Song, "A resolution-like strategy based on a lattice-valued logic", *IEEE Transactions on Fuzzy Systems*, 11 (4): 560-567, 2003.

[22] J.A. Goguen, "The logic of inexact concepts", *Synthese*, 19: 325-373, 1969.

[23] E. Turunen, "Algebraic structures in fuzzy logic", *Fuzzy Sets and Systems*, 52: 181-188, 1992.

[24] J. Pavelka, "On fuzzy logic I: Many-valued rules of inference, II: Enriched residuated lattices and semantics of propositional calculi, III: Semantical completeness of some many-valued propositional calculi", *Zeitschr. F. Math. Logik und Grundlagend. Math*., 25: 45-52, 119-134, 447-464, 1979.

[25] N.C. Ho, W. Wechler, "Hedge algebras: an algebraic approach to structure of sets of linguistic truth values". *Fuzzy Sets and Systems* 35: 281–293, 1990.

[26] N.C. Ho, W. Wechler, "Extended hedge algebras and their application to fuzzy logic". *Fuzzy Sets and Systems* 52: 259–281, 1992.

[27] F. Herrera, L. Martínez, "A 2-tuple fuzzy linguistic representation model for computing with words". *IEEE Trans. Fuzzy Systems*, 8 (6): 746–752 2000.

[28] V. Novak, *Fuzzy Sets and Their Applications*. Philadelphia, PA: Adam Hilger, 1989

[29] S. Lehmke, "On resolution-based theorem proving in propositional fuzzy logic with 'bold' connectives," Ph.D. dissertation, Dept. Comp. Sci., Univ. Dortmund, Dortmund, Germany, 1996.

[30] D. Mundici, N. Olivetti, "Resolution and model building in the infinite-valued valculus of Lukasiewicz". *Theor. Comput. Sci.* 200(1-2): 335-366 (1998)

[31] A. Robinson, A. Voronkov. *Handbook of Automated Reasoning*. MIT Press and Elsevier Science, 2001.