2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Fuzzy Linguistic First Order Logic based on Refined Hedge Algebra

Duc-Khanh Tran
Vietnamese German University
Email: khanh.td@vgu.edu.vn
Minh-Tam Nguyen
Vinh University
Email: nmtam@vinhuni.edu.vn

***Abstract*—We present a fuzzy linguistic first order logic having the truth domain as a refined hedge algebra. The syntax and semantic are defined. Resolution is chosen for the inference system. To capture the approximate nature of resolution inferences, the notion of reliability of resolution inferences is defined. In this respect the resolution procedure can not only prove facts but also indicate how reliable the proof is. We prove the soundness and completeness of the resolution procedure using semantic tree technique.**

***Index Terms*—Refined Hedge Algebra, Linguistic Truth Value, Fuzzy Linguistic First Order Logic, Resolution.**

I. Introduction

Logical inference is an approach to model human reasoning process, which relies on accurate foundations from mathematics. However in many real world applications the reasoning process has to deal with uncertain or fuzzy information. The situation becomes even more complex when the fuzzy information is expressed in natural language. Therefore it is important to build logical systems which are capable of handling logical reasoning with fuzzy linguistic information.

In [5], Zadeh introduced fuzzy set theory and fuzzy logic to deal with uncertain reasoning. Later on, Zadeh [6], [7], [8] used linguistic variables and linguistic values to formalize the reasoning in natural languages. Zadeh's work has played a prominent role in mathematical basis of fuzzy logics, logical foundation of fuzzy inference, and approximate reasoning methods. In [9] Nguyen and Wechler introduced the concept of hedge algebra, which is an algebraic approach to linguistic hedges in Zadeh's fuzzy logic.

The resolution principle, initially designed for two-valued logics by Robinson in 1965 [1], is the heart of many kinds of deductive systems such as theorem proving and logic programming. Finding resolution methods in fuzzy logic as effective as the resolution principle in two-valued logic have been subsequently studied [2], [3], [4].

According to Zadeh [6], [7], [8] the statements *"It is true that Marry is very young'* and *"It is very true that Marry is young"* have approximately equivalent truth values. Therefore inferences in linguistic logics with multiple linguistic variables can be approximately reduced to inferences in a linguistic logic with a single linguistic truth variable, which again can be approximately reduced to inferences in a multi-valued logic with an algebraic truth domain. On the other hand it has been shown that symmetrical refined hedge algebras have a rich enough algebraic structure to be used as a logical truth domain [17]. Indeed the lattice operations join and meet of refined hedge algebras can model the semantics of the logical disjunction and conjunction. Thus linguistic truth terms can be expressed in conjunctive normal form, and the logical formulae become suitable for refutation based inferences such as resolution. This is the starting point for our investigation.

In previous work [10], we proposed a fuzzy propositional logic whose the truth domain is based on a finite symmetrical refined hedge algebra. In this paper we extend the work of [10] from propositional logic to first order logic. We present a fuzzy first order logic having the truth domain generated by a symmetrical refined hedge algebra. We define its syntax and semantics in a similar way as [10]. We give a resolution inference system and prove its soundness and completeness. Soundness means that if a resolution derivation finds the empty clause at some point, then the input set of clauses is unsatisfiable. Completeness means that if the input set of clauses is unsatisfiable then a resolution derivation will find the empty clause after a finite number of inferences. We introduce the notion of reliability as in [10], in order to estimate how reliable a resolution inference is.

Related work

Algebraic and deductive aspects of fuzzy logics have been substantially studied (see e.g., [11] for a detailed survey). In [2] Lee presented a resolution procedure in fuzzy logic and showed the equisatisfiability of logical clauses in fuzzy logics and two-valued logics. Shen et al. [3] went a step further by introducing new concepts such as fuzzy contradictory, contradictory degree, fuzzy resolvent, and confidence of resolvent, and then given a fuzzy refutation complete resolution procedure in which the premises and conclusions, as well as the inferential results are all fuzzy. Smutná and Vojtás [4] gave a sound resolution in many-valued logic with arbitrary connectives and graded information and the resolution truth function to evaluate the truth value of the resolvent of a resolution. Novak [12] gave a logical theory of trichotomous evaluative linguistic expressions. Le et al. [13] introduced the fuzzy linguistic logic programming whose truth domain is

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

based on monotone symmetrical finite hedge algebras, then gave a procedural semantics based on many-valued modus ponens and proved the soundness and completeness of the procedure. Its the soundness and completeness of the algorithm are also proved. Nguyen et al. [14], [15], [16] presented logical systems whose truth domain is based on monotone symmetrical hedge algebras.

Our approach is similar to [3], [14]. The main difference between our work and [3] is that our logic has a algebraic truth domain based on a refined hedge algebra instead of the usual numeric truth domain of fuzzy logic; thus it can directly handle linguistic truth terms of natural languages. Our logic differs from [14] in both algebraic and deductive aspects. We make use of a refined hedge algebra as the truth domain instead of a monotone hedge algebra. This allows us to employ a resolution inference system which handles directly linguistic truth terms in conjunctive normal forms instead of applying hedge moving rules on arbitrary linguistic terms as in [14]. The monotonicity condition is necessary for the soundness of hedge moving rules, while it is not needed for the soundness of our resolution rule. It is worth mentioning that monotonicity is rather too strong a requirement in the context of approximate reasoning in linguistic logics.

Structure of the paper

The paper is organized as follows. Section II introduces refined hedge algebras. Sections III and IV define the syntax and semantics of fuzzy linguistic first order logics. Section V gives a fuzzy linguistic resolution procedure, the soundness and completeness of the procedure are proved. Section VI concludes and draws possible future work.

II. Refined Hedge Algebra

As aforementioned, inferences in natural language can be approximately reduced to inferences in a multi-valued logic with the truth domain based on an algebra of linguistic truth. We will be employing an algebraic truth domain which is a refined hedge algebra. In this section we introduce the most important notions of refined hedge algebras.

We assume the usual notions such as abstract algebra, latice, ... and the notions defined in [9] and [17] such as linguistic variable, hedge, hedge algebra, semantic consistency, PN-homogenity, graded class, unit operation set, ...

We will be working with a class of abstract algebras of the form $AX = (X, G, LH, \leq)$ where X is the term set, G is the set of generators, LH is the set of unary operations representing hedges, and $\leq$ is partial order on X. Furthermore, we assume that X and LH are *semantically consistent*, and LH is *PN-homogeneous*. Accordingly, the set LH is decomposed into two subsets LH^+, LH^- such that $LH^+ + I$ and $LH^- + I$ are finite latices with zero-element I. For c is either $+$ or $-$, SI^c denotes the set of all indexes i which are not single-class elements, LH_i^c denotes graded class i of LH^c, LH^* denotes the set of all strings of hedges in LH. Let us denote by USO (unit operation set) the set of two unit-elements V and S of $LH^+ + I$ and $LH^- + I$. For any element $x \in X$, $LH(x)$ (respectively $LH_i^c[x]$) denotes the set of all elements generated from x by means of hedges in LH (respectively LH_i^c). The set $LH(LH_i^c[x])$ is the union of all the sets $LH(x)$ where $x \in LH_i^c[x]$.

AX is called a *refined hedge algebra*, if X and LH are semantically consistent and the following conditions hold : (where $h, k \in LH$):

(1) Every operation in LH^- is converse to each operation in LH^+.

(2) The unit operation V of $LH^+ + I$ is either positive or negative w.r.t. any operation in LH. In addition, LH should satisfy the PN-homogeneous property.

(3) (Semantic independent property) If u and v are independent, i.e. $u \notin LH(v)$ and $v \notin LH(u)$, then $x \notin LH(v)$ for any $x \in LH(u)$ and vice versa. If $x \neq hx$ then $x \notin LH(hx)$. Further, if $hx \neq kx$ then kx and hx are independent.

(4) (Semantic inheritance) If hx and kx are incomparable, then same for elements $u \in LH(hx)$ and $v \in LH(kx)$. Especially, if a, $b \in G$ and $a < b$ then $LH(a) < LH(b)$. And if $hx < kx$ then

  i) In the case that $h, k \in LH_i^c$, for some $i \in SI^c$ the following statements hold:
  - $\delta hx < \delta kx$, for any $\delta \in LH^*$.
  - δhx and y are incomparable, for any $\forall y \in LH(kx)$ such that $y \not\geq \delta kx$.
  - δkx and z are incomparable, for any $\forall y \in LH(hx)$ such that $z \not\leq \delta hx$.

  ii) Otherwise, $h, k \notin LH_i^c$, then $h'hx \leq k'kx$, for every $h', k' \in UOS$.

(5) (Linear order between the graded classes) Assume that $u \in LH(x)$ and $u \notin LH(LH_i^c[x])$, for any $i \in SI^c$. If exist $v \in LH(hx)$, for $h \in LH_i^c$ such that $u \geq v$ (or $u \leq v$), then $u \geq h'v$ (or $u \leq h'v$) for any $h' \in UOS$.

In the remain of the paper, we will consider a class of refined hedge algebras enjoying properties such as distributivity. symmetry, finiteness, ... which make it suitable to be the truth domain of our logic.

Let $AX = (LH[G], G, LH, \leq, \bot, \mathsf{W}, \top)$ be a refined hedge algebra where

- $G = \{\mathsf{False}, \mathsf{True}\}$,
- $\bot, \top$ are fixed points (i.e. $h\top = \top, h\bot = \bot$ for all $h \in LH$),
- W is the neutral element, and
- $\bot < \mathsf{False} < \mathsf{W} < \mathsf{True} < \top$.

We will be utilizing resolution as the inference system. Resolution works with a special form of formulae called clauses, therefore formulae need to be converted into conjunctive normal form before applying resolution. It is well known that the *distributivity* of the operators of *and* and *or* makes this transformation is feasible. It was shown in [17] that a refined hedge algebra $AX = (X, G, LH, \leq, \bot, \mathsf{W}, \top)$ is a distributive latice when G is a totally ordered set.

Operations of meet and join on AX lattice are determined recursively, based on the corresponding operations on hedge

lattice LH. Denote $LH[x] = \{hx : h \in LH + I\}$, where $x \in X$ and I is the zero-element. It was proved in [17] that $LH[x]$ is a distributive sub-lattice of AX and operations of meet and join on it is defined as follows:

$$hx \vee kx = \begin{cases} (h \cup k)x & \text{if } hx \geq x, \\ (h \cap k)x & \text{if } hx \leq x \end{cases}$$

$$\text{and } hx \wedge kx = \begin{cases} (h \cap k)x & \text{if } hx \geq x, \\ (h \cup k)x & \text{if } hx \leq x \end{cases}$$

where $\cup, \cap$ are the join and meet operators of LH, and $\vee, \wedge$ are the join and meet operators of X.

Let x be an element of the hedge algebra AX and the canonical representation of x is $x = h_n...h_1a$ where $a \in \{\mathsf{True}, \mathsf{False}\}$. The contradictory element of x is an element y such that $y = h_n...h_1a'$ where $a' \in \{\mathsf{True}, \mathsf{False}\}$ and $a' \neq a$, denoted by $\overline{x}$. The contradictory element of $\top$ is $\bot$ and conversely. The contradictory element of W is itself. AX is said to be a *symmetrical refined hedge algebra* if every element $x \in X$ has a unique contradictory element in X.

In practice, natural language only uses finite string of hedges. Therefore it is useful to limit the set X to consists of only finite length elements. In this case we say that the symmetrical refined hedge algebra is finite. Let $LH_p[G] = \{h_n...h_1a : h_i \in LH + I, a \in G, n \leq p\}$, then a *finite symmetrical refined hedge algebra* $AX = (LH_p[G], G, LH, \leq, \bot, \mathsf{W}, \top)$ is a complete distributive lattice.

Let x and y be two elements of the finite symmetrical refined hedge algebra AX, then

- the negation operator is an unary operator, which is defined by $\neg x = \overline{x}$, where $\overline{x}$ is the contradictory element of x,
- the implication operator is a binary operator, which is defined through negative and join operators: $x \to y = \neg x \vee y$.

Theorem II.1. *[17] Let $AX = (LH_p[G], G, LH, \leq, \bot, \mathsf{W}, \top, \neg, \vee, \wedge, \to, \bot, \mathsf{W}, \top)$ be a finite symmetrical refined hedge algebra. Then, for all $x, y \in LH_p[G]$, for all $h, k \in LH$, we have:*

1) $\neg(hx) = h\neg x$
2) $\neg(\neg x) = x$
3) $\neg(x \vee y) = \neg x \wedge \neg y$ *and* $\neg(x \wedge y) = \neg x \vee \neg y$
4) $x \wedge \neg x \leq y \vee \neg y$
5) $x \wedge \neg x \leq \mathsf{W} \leq x \vee \neg x$
6) $\neg\top = \bot, \neg\bot = \top$ *and* $\neg\mathsf{W} = \mathsf{W}$
7) $x > y$ *iff* $\neg x < \neg y$
8) $x \to y = \neg y \to \neg x$
9) $x \to (y \to z) = y \to (x \to z)$
10) $x \to y \leq x' \to y'$ *if* $x \leq x'$ *and* $y \geq y'$
11) $x \to y = \top$ *iff* $x = \bot$ *or* $y = \top$
12) $\top \to x = x$ *and* $x \to \top = \top$; $\bot \to x = \top$ *and* $x \to \bot = \neg x$
13) $x \to y \geq \mathsf{W}$ *iff* $x \leq \mathsf{W}$ *or* $y \geq \mathsf{W}$, *and* $x \to y \leq \mathsf{W}$ *iff* $x \geq \mathsf{W}$ *or* $y \leq \mathsf{W}$

III. Syntax

The syntactical component of our logic is defined as in first order logic with the usual syntactical notions such as alphabet, term, formulae, ... for our logic.

Definition III.1. *An alphabet consists of the followings:*

- *variables:* $x, y, z, \ldots$;
- *function symbols: a set FS of symbols $f, g, h, \ldots$ each of n-arity ($n \geq 0$); a function symbol with 0-arity is called a constant;*
- *predicate symbols: a set PS of symbols $P, Q, R, \ldots$ each of n-arity ($n \geq 0$); predicate symbol with 0-arity is called a logical constant symbol;*
- *logical connectives:* $\vee, \wedge, \neg, \to, \leftrightarrow$;
- *quantifiers:* $\forall, \exists$;
- *auxiliary symbols:* $\square, (,), \ldots$;

Definition III.2. *A term is defined recursively as follows:*

- *either a constant symbol or a variable is a term,*
- *if f is a n-ary function symbol and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ ($n > 0$) is a term.*

Definition III.3. *An atom is either a 0-ary predicate symbol or a n-ary predicate symbol $P(t_1, \ldots, t_n)$ ($n > 0$), where $t_1, \ldots, t_n$ are terms. variables.*

Definition III.4. *Let A be an atom and α be a logical constant symbol. Then A^α is called a literal.*

Definition III.5. *Formulae are defined recursively as follows:*

- *a literal is a formula,*
- *if ϕ, ψ are formulae, then $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \to \psi$, $\phi \leftrightarrow \psi$, $\neg\phi$ are formulae, and*
- *if x is a variable and ϕ is a formula then $\forall x\phi, \exists x\phi$ are formulae.*

A variable bounded to a quantifier is called a bounded variable. A free variable is a variable which is not bounded to any quantifiers.

An expression is either a term or an atom or a formula. An expression is ground if it does not contain any variables.

A clause is a finite set of literals, is usually written as a disjunction $l_1 \vee l_2 \vee ... \vee l_n$, where l_i is a literal (for $i = 1, 2, ...$). An empty clause is denoted by $\square$. A formula is said to be in conjunctive normal form (CNF) if it is a conjunction of clauses.

A substitution is a finite set of specifications of the form $[t/v]$ in which t is a term and v is a variable. Substitutions are usually written in set notation: $\{t_1/v_1, t_2/v_2, .., t_n/v_n\}$. A substitution $\{t_1/v_1, t_2/v_2, .., t_n/v_n\}$ is ground if all terms $t_1, \ldots, t_n$ are ground. The product of two substitutions σ and θ, denoted by $\sigma o\theta$, is defined such that if $x_i\sigma = y_i$ and $y_i\theta = s_i$ then $x_i\sigma o\theta = s_i$.

Let $\theta = \{t_1/v_1, t_2/v_2, .., t_n/v_n\}$ be a substitution and e be a expression. An instance $e\theta$ of e is the expression obtained by replacing simultaneously all occurrences of variables $v_1, \ldots, v_n$ with the corresponding terms $t_1, \ldots, t_n$.

Let e_1, e_2 be expressions, and γ be a substitution. Then γ is called a unifier e_1 and e_2 if $e_1 = e_2\gamma$. We say that a unifier γ is more general than another unifier σ if there exists an substitution ϕ such that $\sigma = \gamma o \phi$. We say that γ is the most general unifier (m.g.u for short) if there is no unifier more general than γ.

IV. Semantics

We define the usual semantic notions for our logic such as structure, interpretation, evaluation, model, satisfiability, ... in a similar way as for first order logic, except that we employ the following finite symmetrical refined hedge algebra for the truth domain

$$AX = (LH_p[G], G, LH, \leq, \bot, \mathsf{W}, \top, \neg, \vee, \wedge, \rightarrow)$$

Definition IV.1. *A structure* M *of an alphabet* $\mathcal{A}$ *is a triplet* (D, I^f, I^p)*, where:*

- D *is a nonempty set called the domain of the structure, and:*
  - $\mathcal{F}_D$ *is the set of functions on* D*:* $\mathcal{F}_D = \{f_D | f_D : D^n \longrightarrow D, n > 0\}$
  - $\mathcal{P}_D$ *is the set of relations on* D*:* $\mathcal{P}_D = \{P_D | P_D : D^n \longrightarrow X, n \geq 0\}$*, where* $X = LH_p[G] \cup \{\bot, \mathsf{W}, \top\}$
- $I^f : FS \longrightarrow \mathcal{F}_D$*, for every n-ary function symbol* f*,* f *is assigned to an element in* $\mathcal{F}_D$
- $I^p : PS \longrightarrow \mathcal{R}_D$*, for every n-ary predicate symbol* P*,* P *is assigned to an element in* $\mathcal{P}_D$

Definition IV.2. *An assignment* σ *is a mapping of the set of variables* V *into the domain* D*,* $\sigma : V \longrightarrow D$*.*

Definition IV.3. *An interpretation* $\mathcal{I}$ *is a pair* (M, σ)*, where* M *is a structure and* σ *is an assignment.*

The evaluation of a term t under the interpretation $\mathcal{I}$ is denoted by $[|t|]^{\mathsf{M}}_{\sigma}$, or $\mathcal{I}(t)$. Likewise, the evaluation of a formula ϕ under the interpretation $\mathcal{I}$ is denoted by $[|\phi|]^{\mathsf{M}}_{\sigma}$, or $\mathcal{I}(\phi)$.

Definition IV.4. *The evaluation of a term under an interpretation* $\mathcal{I} = (\mathsf{M}, \sigma)$ *is determined as follows:*

- $[|c|]^{\mathsf{M}} = c^{\mathsf{M}}$ *for a constant* c
- $[|x|]^{\mathsf{M}}_{\sigma} = \sigma(x)$ *for a variable* x
- $[|f(t_1, \ldots, t_n)|]^{\mathsf{M}}_{\sigma} = f^{\mathsf{M}}([|t_1|]^{\mathsf{M}}_{\sigma}, \ldots, [|t_n|]^{\mathsf{M}}_{\sigma})$ *for a term* $f(t_1, \ldots, t_n)$

Definition IV.5. *Let* $\mathcal{I} = (\mathsf{M}, \sigma)$ *be an interpretation and* A *be an atom such that* $[|A|]^{\mathsf{M}}_{\sigma} = \alpha^{\mathsf{M}}_1$*. The evaluation of a literal* A^{α_2} *under the interpretation* $\mathcal{I} = (\mathsf{M}, \sigma)$ *is determined as follows:*

$$[|A^{\alpha_2}|]^{\mathsf{M}}_{\sigma} = \begin{cases} \alpha^{\mathsf{M}}_1 \wedge \alpha^{\mathsf{M}}_2 \text{ if } \alpha^{\mathsf{M}}_1, \alpha^{\mathsf{M}}_2 > \mathsf{W}, \\ \neg(\alpha^{\mathsf{M}}_1 \vee \alpha^{\mathsf{M}}_2) \text{ if } \alpha^{\mathsf{M}}_1, \alpha^{\mathsf{M}}_2 \leq \mathsf{W}, \\ (\neg\alpha^{\mathsf{M}}_1) \vee \alpha^{\mathsf{M}}_2, \text{ if } \alpha^{\mathsf{M}}_1 > \mathsf{W}, \alpha^{\mathsf{M}}_2 \leq W, \text{ and} \\ \alpha^{\mathsf{M}}_1 \vee (\neg\alpha^{\mathsf{M}}_2), \text{ if } \alpha^{\mathsf{M}}_1 \leq \mathsf{W}, \alpha^{\mathsf{M}}_2 > \mathsf{W}. \end{cases}$$

Definition IV.6. *The evaluation of formulae under an interpretation* $\mathcal{I} = (\mathsf{M}, \sigma)$ *is determined recursively as follows:*

- $[|\neg\phi|]^{\mathsf{M}}_{\sigma} = \neg[|\phi|]^{\mathsf{M}}_{\sigma}$
- $[|\phi \wedge \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \wedge [|\varphi|]^{\mathsf{M}}_{\sigma}$
- $[|\phi \vee \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \vee [|\varphi|]^{\mathsf{M}}_{\sigma}$
- $[|\phi \rightarrow \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \rightarrow [|\varphi|]^{\mathsf{M}}_{\sigma}$
- $[|\phi \leftrightarrow \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \leftrightarrow [|\varphi|]^{\mathsf{M}}_{\sigma}$
- $[|\exists x\phi|]^{M}_{\sigma} = max_{c \in D}\{[|\phi|]^{M}_{\sigma'} | \sigma' = \sigma \cup \{x := c\}\}$
- $[|\forall x\phi|]^{M}_{\sigma} = min_{c \in D}\{[|\phi|]^{M}_{\sigma'} | \sigma' = \sigma \cup \{x := c\}\}$

Definition IV.7. *Let* $\mathcal{I} = (\mathsf{M}, \sigma)$ *be an interpretation and* ϕ *be a formula.*

- ϕ *is said to be true under the interpretation* $\mathcal{I}$*, or* $\mathcal{I}$ *satisfies* ϕ*, or* $\mathcal{I}$ *is a model of* ϕ*, iff* $[|\phi|]^{\mathsf{M}}_{\sigma} > \mathsf{W}$*, denoted by* $\mathcal{I} \models \phi$*.*
- *We say that* ϕ *is satisfiable iff it has a model.*
- *We say that* ϕ *is unsatisfiable iff it has no model.*
- ϕ *is said to be false under the interpretation* $\mathcal{I}$*, or* $\mathcal{I}$ *falsifies* ϕ*, iff* $[|\phi|]^{\mathsf{M}}_{\sigma} \leq \mathsf{W}$*.*
- ϕ *is valid in the structure* M *if* $[|\phi|]^{\mathsf{M}}_{\sigma} > \mathsf{W}$ *for all assignments* σ*, denoted by* $\mathsf{M} \models \phi$*.* M *is called a model of* ϕ*.*
- ϕ *is said to be tautology iff it is valid in all structures* M*, denoted by* $\models \phi$*.*

Definition IV.8. *Let* $\mathcal{I} = (\mathsf{M}, \sigma)$ *be an interpretation, let* ϕ *and* φ *be formulae. Then*

- φ *is a logical consequence of* ϕ*, denoted by* $\phi \models \varphi$*, iff for every interpretation* $\mathcal{I}$*,* $\mathcal{I} \models \phi$ *implies that* $\mathcal{I} \models \varphi$*.*
- ϕ *and* φ *are said to be logically equivalent, denoted by* $\phi \equiv \varphi$*, iff* $\phi \models \varphi$ *and* $\varphi \models \phi$*.*
- ϕ *and* φ *are said to be equisatisfiable if* ϕ *is satisfiable iff* φ *is satisfiable.*

Theorem IV.1. *Let* ϕ, φ *and* ψ *be formulas. Then the following properties hold:*

1) *Idempotency:*
   - $\phi \vee \phi \equiv \phi$
   - $\phi \wedge \phi \equiv \phi$
2) *Implication:*
   - $\phi \rightarrow \varphi \equiv (\neg\phi) \vee \varphi$
   - $(\phi \equiv \varphi) \equiv (\phi \rightarrow \varphi) \wedge (\varphi \rightarrow \phi)$
3) *Double negation:*
   - $\neg\neg\phi \equiv \phi$
4) *De Morgan:*
   - $\neg(\phi \vee \varphi) \equiv (\neg\phi) \wedge (\neg\varphi)$
   - $\neg(\phi \wedge \varphi) \equiv (\neg\phi) \vee (\neg\varphi)$
5) *Commutativity:*
   - $\phi \vee \varphi \equiv \varphi \vee \phi$
   - $\phi \wedge \varphi \equiv \varphi \wedge \phi$
6) *Associativity:*
   - $\phi \vee (\varphi \vee \psi) \equiv (\phi \vee \varphi) \vee \psi$
   - $\phi \wedge (\varphi \wedge \psi) \equiv (\phi \wedge \varphi) \wedge \psi$
7) *Distributivity:*
   - $\phi \vee (\varphi \wedge \psi) \equiv (\phi \vee \varphi) \wedge (\phi \vee \psi)$
   - $\phi \wedge (\varphi \vee \psi) \equiv (\phi \wedge \varphi) \vee (\phi \wedge \psi)$
8) *Quantifier:*

- $(Qx\alpha) \wedge \beta \equiv Qx(\alpha \wedge \beta)$
- $(Qx\alpha) \vee \beta \equiv Qx(\alpha \vee \beta)$
- $\alpha \wedge (Qx\beta) \equiv Qx(\alpha \wedge \beta)$
- $\alpha \vee (Qx\beta) \equiv Qx(\alpha \vee \beta)$

where $Q \in \{\exists, \forall\}$

A formula φ is said to be in conjunctive normal form (CNF) if φ is a conjunction of clauses. Below is a rule-based algorithm to transform an arbitrary formula into an equisatisfiable CNF formula. The algorithm consists of seven steps:

1) Eliminate implications and equivalences:
   $\alpha \to \beta \Longrightarrow_{CNF} \neg\alpha \vee \beta$
   $\alpha \leftrightarrow \beta \Longrightarrow_{CNF} (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha)$
2) Move negations inward:
   $\neg\neg\alpha \Longrightarrow_{CNF} \alpha$
   $\neg\exists x\alpha \Longrightarrow_{CNF} \forall x\neg\alpha$
   $\neg\forall x\alpha \Longrightarrow_{CNF} \exists x\neg\alpha$
   $\neg(\alpha \vee \beta) \Longrightarrow_{CNF} \neg\alpha \wedge \neg\beta$
   $\neg(\alpha \wedge \beta) \Longrightarrow_{CNF} \neg\alpha \vee \neg\beta$
3) Standardize variables: if two variables have the same name but are in two different clauses then rename one of them.
4) Move quantifiers outward:
   $(Qx\alpha) \wedge \beta \Longrightarrow_{CNF} Qx(\alpha \wedge \beta)$
   $(Qx\alpha) \vee \beta \Longrightarrow_{CNF} Qx(\alpha \vee \beta)$
   $\alpha \wedge (Qx\beta) \Longrightarrow_{CNF} Qx(\alpha \wedge \beta)$
   $\alpha \vee (Qx\beta) \Longrightarrow_{CNF} Qx(\alpha \vee \beta)$
   where $Q \in \{\exists, \forall\}$
5) Eliminate existential quantifiers:
   $\forall x_1 \ldots \forall x_n \exists x\alpha \Longrightarrow_{CNF} \forall x_1 \ldots \forall x_n \alpha[x := \pi_{(x_1,\ldots,x_n)}]$
   where π is a n-ary function symbol, also called "Skolem function"
6) Eliminate universal quantifiers: $\forall x\alpha \Longrightarrow_{CNF} \alpha$
7) Distribute disjunctions inward over conjunctions:
   $\alpha \wedge (\beta \vee \gamma) \Longrightarrow_{CNF} (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
   $\alpha \vee (\beta \wedge \gamma) \Longrightarrow_{CNF} (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Lemma IV.1. *Let ϕ and φ be formulae such that $\phi \Longrightarrow_{CNF} \varphi$. Then ϕ and φ are equisatisfiable.*

The following theorem follows from Lemma IV.1.

Theorem IV.2. *Any formula of arbitrary form can be converted into an equisatisfiable CNF formula.*

As in two-valued first order logic, for deciding satisfiability of formulae it is sufficient to consider a special class of interpretations, namely Herbrand's interpretations, as satisfiability can be reduced to satisfiability in Herbrand's interpretation.

Definition IV.9. *The Herbrand universe, or H-universe for short, of an alphabet $\mathcal{A}$, denoted by $U(\mathcal{A})$, is the set of all ground terms built over $\mathcal{A}$.*

Definition IV.10. *The set of all ground atoms built over $\mathcal{A}$ is called the Herbrand base, or H-base for short, denoted by $B(\mathcal{A})$.*

Definition IV.11. *A Herbrand structure, or H-structure for short, of an alphabet $\mathcal{A}$ is a structure having the domain which is the Herbrand universe $U(\mathcal{A})$.*

Definition IV.12. *A Herbrand interpretation, or H-interpretation for short, of an alphabet $\mathcal{A}$ is a pair (M_H, σ_H), where M_H is a H-structure of the alphabet $\mathcal{A}$ and $\sigma_H : V \longrightarrow U(\mathcal{A})$ is a variable assignment.*

It is convenient to define the notions of H-universe, H-base, H-structure and H-interpretation for a set of clauses S. Let S be a clause set, let $\mathcal{A}(S)$ be the alphabet containing exactly the constant symbols, function symbols, predicate symbols appearing in S, along with usual symbols such as variables, logical symbols, auxiliary symbols. The H-universe of S, denoted by $U(S)$, is the H-universe of the alphabet $\mathcal{A}(S)$. The H-base of S, denoted by $B(S)$, is the H-base of the alphabet $\mathcal{A}(S)$. An H-structure of S is an H-structure of the alphabet $\mathcal{A}(S)$. An H-interpretation of S is an H-interpretation of the alphabet $\mathcal{A}(S)$.

Theorem IV.3. *A clause set S is satisfiable iff S is satisfied in an H-interpretation.*

Proof: ($\Rightarrow$) Assume that S is satisfiable, then it is satisfied by an interpretation $\mathcal{I} = (\mathsf{M}, \sigma)$ over the domain D. We construct an H-interpretation of S based on the existing interpretation $\mathcal{I}$ as follows: $\mathcal{H} = (\mathsf{M}_H, \sigma_H)$

- $\mathsf{M}_H = (U(S)), H^f, H^p)$, where:
  - $U(S)$ is the H-universe of S,
  - $H^f(f)(h_1, \ldots, h_n) = f(h_1, \ldots, h_n) \in U(S))$,
  - $H^p(P)(h_1, \ldots, h_n) = I^p(P)(\mathcal{I}(h_1), \ldots, \mathcal{I}(h_n)) \in X$.
- $\sigma_H(x) = h$ where $h \in U(S))$ and $\mathcal{I}(h) = \sigma(x)$.

Then, we have:

$$\begin{aligned}\mathcal{H}(P(t_1, \ldots, t_n)) &= H^p(P)(\sigma_H(t_1), \ldots, \sigma_H(t_n)) \\ &= I^p(P)(\mathcal{I}(\sigma_H(t_1)), \ldots, \mathcal{I}(\sigma_H(t_n))) \\ &= I^p(P)(\sigma(t_1), \ldots, \sigma(t_n)) \\ &= \mathcal{I}(P(t_1, \ldots, t_n))\end{aligned}$$

This shows that if S is true under $\mathcal{I}$, then it is true under $\mathcal{H}$ as well.

($\Leftarrow$) If S is satisfied under an H-interpretation $\mathcal{H}$, then it is obvious that S is satisfiable. ■

V. Resolution

In fuzzy linguistic logic the degree of contradiction can vary because the truth domain contains more than two elements. For instance sets of formulae $\{A^{\mathsf{VeryTrue}}, A^{\mathsf{VeryFalse}}\}$ is "more contradictory" than the set formulae$\{A^{\mathsf{LessTrue}}, A^{\mathsf{LessFalse}}\}$. As in [10], the notion of reliability is needed to capture the fuzziness of resolution inferences.

Let α be an element of X such that $\alpha > \mathsf{W}$ and C be a clause. The clause C with its reliability α is the pair (C, α). The reliability α of a clause set $S = \{C_1, C_2, \ldots, C_n\}$, is

defined as follows:

$$\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$$

where α_i is the reliability of C_i (for $(i = 1, 2, \ldots, n)$).

An inference rule R working with clauses with reliabilities is represented as follows:

$$\frac{(C_1, \alpha_1) \quad (C_2, \alpha_2) \ \ldots \ (C_n, \alpha_n)}{(C, \alpha)}$$

where $C_1, \alpha_1) \quad (C_2, \alpha_2) \ \ldots \ (C_n, \alpha_n)$ are premises and (C, α) is the conclusion. We say that α is the reliability of R, provided that $\alpha \leq \alpha_i$ for $i = 1, 2, \ldots, n$.

An inference is sound if its conclusion is a logical consequence of its premises. That is, for any interpretation $\mathcal{I}$, if $\mathcal{I}(C_1 \wedge C_2 \wedge \ldots \wedge C_n) > \mathsf{W}$ then $\mathcal{I}(C) > \mathsf{W}$.

Definition V.1. *Define the resolution rule as follows:*

$$\frac{(A^a \vee B^b, \alpha_1) \quad (C^c \vee D^d, \alpha_2)}{((A^a \vee D^d)\gamma, \alpha_3)}$$

where b, c and α_3 satisfy the following conditions:

$$\begin{cases} b \wedge c \leq \mathsf{W} \\ b \vee c > \mathsf{W} \\ \gamma \text{ is the most general unifier of } B \text{ and } C \\ \alpha_3 = f(\alpha_1, \alpha_2, b, c) \end{cases}$$

with f is a function such that $\alpha_3 \leq \alpha_1$, and $\alpha_3 \leq \alpha_2$.

$((A^a \vee D^d)\gamma, \alpha_3)$ is called a resolvent of $(A^a \vee B^b, \alpha_1)$ and $(C^c \vee D^d, \alpha_2)$.

In Definition V.1 the reliability α_3 is defined so as to be smaller or equal to both α_1 and α_2. This makes sure that the more inferences we need to deduce a clause the less reliable the obtained clause is. There are different ways to define α_3. Below we define α_3 in three ways based on $\wedge$ and $\vee$ operators.

The reliability of the resolution rule V.1 based on $\wedge$ operator is given by:

$$\alpha_3 = f(\alpha_1, \alpha_2, b, c) = \alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c)$$

The reliability of the resolution rule V.1 based on $\vee$ operator is given by:

$$\alpha_3 = f(\alpha_1, \alpha_2, b, c) = \alpha_1 \wedge \alpha_2 \wedge (b \vee c)$$

The reliability of the resolution rule V.1 based on the combination of $\wedge$ and $\vee$ operators is given by:

$$\alpha_3 = f(\alpha_1, \alpha_2, b, c) = \alpha_1 \wedge \alpha_2 \wedge (\neg(b \wedge c)) \wedge (b \vee c)$$

Proposition V.1. *The reliability based on $\wedge$ operator (respectively $\vee$ operator or the combination of $\wedge$ and $\vee$ operators) satisfies the conditions on α_3 in Definition V.1.*

Proof: For $\wedge$ operator we need to prove that $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) \leq \alpha_1$ and $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) \leq \alpha_2$ and $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) > \mathsf{W}$. By definition of $\wedge$ it is clear that that $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) \leq \alpha_1$ and $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) \leq \alpha_2$. Additionally, we have that $\alpha_1, \alpha_2 > \mathsf{W}$. Moreover, $b \wedge c \leq \mathsf{W}$ implies $\neg(b \wedge c) > \mathsf{W}$. Then $\alpha_1 \wedge \alpha_2 \wedge \neg(b \wedge c) > \mathsf{W}$.

For $\vee$ operator we need to prove that $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) \leq \alpha_1$ and $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) \leq \alpha_2$ and $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) > \mathsf{W}$. By definition of $\wedge$ it is clear that $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) \leq \alpha_1$ and $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) \leq \alpha_2$. Additionally, we have that $\alpha_1, \alpha_2 > \mathsf{W}$ and $b \wedge c \leq \mathsf{W}$. Therefore $\alpha_1 \wedge \alpha_2 \wedge (b \vee c) > \mathsf{W}$.

For the combination operator, we have

$$\begin{aligned} \alpha_3 &= \alpha_1 \wedge \alpha_2 \wedge (\neg(b_1 \wedge b_2)) \wedge (b_1 \vee b_2) \\ &= (\alpha_1 \wedge \alpha_2 \wedge \neg(b_1 \wedge b_2)) \wedge (\alpha_1 \wedge \alpha_2 \wedge (b_1 \vee b_2)) \end{aligned}$$

Clearly $\alpha_3 \leq \alpha_1$ and $\alpha_3 \leq \alpha_2$. Applying the results of $\wedge$ and $\vee$ operators we have $\alpha_3 > \mathsf{W}$. ■

Theorem V.1. *The resolution rule V.1 is sound.*

Proof: Let $\mathcal{I} = (\mathsf{M}, \sigma)$ be an interpretation. We need to prove that if $\mathcal{I}((A^a \vee B^b) \wedge (C^c \vee D^d)) > \mathsf{W}$ then $\mathcal{I}((A^a \vee D^d)\gamma) > \mathsf{W}$.

$\mathcal{I}((A^a \vee B^b) \wedge (C^c \vee D^d)) > \mathsf{W}$ implies that $\mathcal{I}((A^a \vee B^b) \wedge (C^c \vee D^d\gamma)) > \mathsf{W}$.

$$\begin{aligned} &\mathcal{I}(((A^a \vee B^b) \wedge (C^c \vee D^d))\gamma) \\ &= \mathcal{I}(((A^a \wedge C^c) \vee (A^a \wedge D^d) \vee (B^b \wedge C^c) \vee (B^b \wedge D^d))\gamma) \\ &= \mathcal{I}((A^a \wedge C^c)\gamma \vee (A^a \wedge D^d)\gamma \vee (B^b \wedge C^c)\gamma \vee (B^b \wedge D^d)\gamma) \\ &= \mathcal{I}((A^a \wedge C^c)\gamma) \vee \mathcal{I}((A^a \wedge D^d)\gamma) \vee \mathcal{I}((B^b \wedge C^c)\gamma) \vee \\ &\quad \mathcal{I}((B^b \wedge D^d)\gamma) \end{aligned}$$

It is easy to see that:

- $\mathcal{I}((A^a \wedge C^c)\gamma) \leq \mathcal{I}(A^a\gamma) \leq \mathcal{I}((A^a \vee D^d)\gamma)$,
- $\mathcal{I}((A^a \wedge D^d)\gamma) \leq \mathcal{I}((A^a \vee D^d)\gamma)$,
- $\mathcal{I}((B^b \wedge C^c)\gamma) \leq \mathsf{W}$, and
- $\mathcal{I}((B^b \wedge D^d)\gamma) \leq \mathcal{I}(D^d\gamma) \leq \mathcal{I}((A^a \vee D^d)\gamma)$.

This means that if $\mathcal{I}((A^a \vee D^d)\gamma \leq \mathsf{W}$ then $\mathcal{I}((A^a \wedge C^c)\gamma) \vee \mathcal{I}((A^a \wedge D^d)\gamma) \vee \mathcal{I}((B^b \wedge C^c)\gamma) \vee \mathcal{I}((B^b \wedge D^d)\gamma) \leq \mathsf{W}$, or equivalently $\mathcal{I}((A^a \vee B^b) \wedge (C^c \vee D^d)) \leq \mathsf{W}$, which contradicts with the fact that $\mathcal{I}((A^a \vee B^b) \wedge (C^c \vee D^d)) > \mathsf{W}$. This complete the proof of the theorem. ■

Definition V.2. *Define the factoring rule as follows:*

$$\frac{(A^a \vee B^a \vee C^c, \alpha)}{((A^a \vee C^c)\gamma, \alpha)}$$

where γ is the most general unifier of A and B.

$((A^a \vee C^c)\gamma, \alpha)$ is called a factor of $(A^a \vee B^b \vee C^c, \alpha)$.

Theorem V.2. *The factoring rule V.2 is sound.*

Proof: Straightforward. ■

Definition V.3. *A resolution derivation is a sequence of the form*

$$S_0, \ldots, S_i, \ldots$$

where

- *S_i is a set of clauses with reliability (for $i = 1, \ldots, n$), and*
- *$S_{i+1} = S_i \cup (C, \alpha)$, and $(C, \alpha) \notin S_i$, and (C, α) is the conclusion of a resolution inference with premises from S_i or of a factoring with premisse from S_i.*

Below we are going to prove the soundness and completeness of resolution derivations.

Theorem V.3 (Soundness)**.** *Let $S_0, \ldots, S_i, \ldots$ be a resolution derivation. If S_n contains the empty clause, (for some $n = 0, 1, ...$) , then S_0 is unsatisfiable.*

Proof: By Theorems V.1 and V.2, S_i and S_{i+1} are logically equivalent. Therefore if S_n contains the empty clause, meaning that S_n is unsatisfiable, then S_0 is unsatisfiable ■

To prove the completeness of the resolution derivation, we use semantic tree method. According to Theorem IV.3, instead of considering all possible interpretations of a clause set, we only consider its H-interpretations and build semantic trees based on the H-interpretations.

Definition V.4. *Let S be a set of clauses and $B(S)$ be the H-base of S. A semantic tree of S is a n-level complete binary tree constructed as follows:*

- *Each level corresponds to an element of $B(S)$. If the i^{th} level corresponds to the atom $A_i \in B(S)$, then the left edge of each node at the level i is assigned with the label $A_i \leq \mathsf{W}$, and the right edge of each node at the level i is assigned with the label $A_i > \mathsf{W}$,*
- *Each element of $B(S)$ corresponds to exactly one level in the tree, which means that if $A_i \in A(S)$ appears in level i then it must not be chosen in any other levels.*

Notice that each path from the root to a certain leaf in a semantic tree corresponds to an H-interpretation of the clause set S. The tree's depth is infinite if $B(S)$ is infinite. There are different semantic trees for a given set of clauses, depending on choices of atoms in each level on the tree.

Let T be a semantic tree of a set of clauses S. A clause C of S is failed at node N of T if there exist an H-interpretation $\mathcal{I}$ corresponding to a branch of T containing N such that C is false under $\mathcal{I}$. A node N of T is called a failure node of C iff C is failed at N but is not failed at any nodes above N. A node N of T is an inference node if both of its successor nodes are failure nodes. If every branch in T contains a failure node, cutting off its descendants from T, we have a tree T' which is called a closed tree of S; if the number of nodes in T' is finite then T' is called a finite closed tree.

Lemma V.1. *There always exists an inference node on finite closed tree.*

Proof: Assume that we have a closed tree T . Because T has finite level, so there exists one (or more) leaf node on T at the highest level, let say this node is called j. Let i be parent node of j. By definition of closed tree, i cannot be failure node. Therefore, i has another child node, named k. If k is a failure node then i is inference node, the lemma is proved. If k is not a failure node then it has two child nodes: l, m. Clearly l, m are at higher level than j. This contradicts with the assumption that j is at the highest level. Therefore k is a failure node and i is an inference node. This completes the proof of the theorem." ■

Next, we shall recall König's lemma.

Lemma V.2 (König's Lemma)**.** *Suppose that T is an infinite finitely branching tree. Then there exists an infinite branch B through T.*

A corollary of König's lemma is that if every branch of a tree T has finite depth, then the number of nodes of T is finite.

Theorem V.4. *Let S be a clause set. S is unsatisfiable iff for every semantic tree of S, there exists a finite closed tree.*

Proof: ($\Rightarrow$) Suppose that S is unsatisfiable and T is a semantic tree of S. Let B be a branch of T, we denote $\mathcal{I}_B$ the H-interpretation corresponding to B. By Theorem IV.3, $\mathcal{I}_B$ falsifies S. Thus, there exists a ground instance C' of a clause C in S which is false under $\mathcal{I}_B$. Therefore there exists a failure node N_B on the branch B. Since C' has a finite number of literals, N_B is a finite number of edges away from the root. We have actually shown that there is a failure node on every branch of T which is a finite number of edges away from the root. The tree T' is obtained by T removing all nodes which are below the failure node. T' is a closed tree. Every branch of T' has finite length. By König's lemma, T' has finite nodes.

($\Leftarrow$) Assume that there is always a finite closed tree for a semantic tree T of the set of clauses S. Then every branch of T contains a failure node, it means any H-interpretation falsifies S. By Theorem IV.3, S is unsatisfiable. ■

Lemma V.3 (Lifting lemma)**.** *Let C be a resolvent of $\{C_1, C_2\}$ and C_1', C_2' be instances of C_1, C_2 respectively. If C' is a resolvent of $\{C_1', C_2'\}$ then C' is an instance of C (or of a factor of C).*

Proof: $C_1' = {\Gamma_1'}^{\alpha} \vee {T_1'}^{\beta_1}, C_2' = {\Gamma_2'}^{\delta} \vee {T_2'}^{\beta_2}$, γ is a m.g.u of T_1', T_2'. Let σ be a substitution such that $C_1' = C_1\sigma, C_2' = C_2\sigma$ and $C_1 = {\Gamma_1}^{\alpha} \vee {T_1}^{\beta_1}, C_2 = {\Gamma_2}^{\delta} \vee {T_2}^{\beta_2}$. By resolution rule V.1, $C' = ({\Gamma_1'}^{\alpha} \vee {\Gamma_2'}^{\delta})\gamma = ({\Gamma_1}^{\alpha} \vee {\Gamma_2}^{\delta})\gamma o\sigma$ because $\Gamma_1' = \Gamma_1\sigma, \Gamma_2' = \Gamma_2\sigma$. Assume that θ is the m.g.u of T_1, T_2 then θ is more general than γ, which implies that θ is more general than $\gamma o\sigma$. Thus $C' = ({\Gamma_1}^{\alpha} \vee {\Gamma_2}^{\delta})\gamma o\sigma$ is an instance of $C = ({\Gamma_1}^{\alpha} \vee {\Gamma_2}^{\delta})\theta$ (or of a factor of $C = ({\Gamma_1}^{\alpha} \vee {\Gamma_2}^{\delta})\theta$). ■

Theorem V.5 (Completeness)**.** *Let $S_0, \ldots, S_i, \ldots$ is a resolution derivation. If S_0 is unsatisfiable then there exists S_k containing the empty clause.*

Proof: According to Theorem V.4 if S_0 is unsatisfiable, then for every semantic tree T_0 of S_0 there is a corresponding finite closed tree T_0'. By Lemma V.1, there exists an inference node N on T_0'. Let S_0' be the set of all ground instance of clauses in S_0 Let $C_1', C_2' \in S_0'$ be the ground instances of two clauses $C_1, C_2 \in S_0$ such that C_1', C_2' are failed at the two children of N. Assume that the level of N corresponds to a ground atom L'. Then C_1' and C_2' contains the literal ${L'}^{\alpha_1}$ and ${L'}^{\alpha_2}$ where $\alpha_1 > \mathsf{W}$ and $\alpha_2 \leq \mathsf{W}$.

Resolving C_1' and C_2', we obtain the clause C' not containing L', therefore C' is failed the node N. By Lemma V.3, we can find a resolvent C of $C_1, C_2 \in S_0$ such that C' is an

instance of C, or of a factor of C. The closed semantic tree T_1' associated to $S_1 = S_0 \cup C$ has fewer nodes than T_0'.

The process is then iterated. Because T_0' has a finite number of nodes so there exists k such that T_k' of S_k consists only of one root node, then $\Box$ must be in S_k'. By Lemma V.3, S_k contains $\Box$. This completes the proof of the theorem. ■

Let us consider an example to illustrate how a resolution derivation works.

Example V.1. *Let* $AX = (LH_p[G], G, LH, \leq, \bot, \mathsf{W}, \top, \neg, \vee, \wedge, \rightarrow)$ *be a finite symmetrical refined hedge algebra where* $G = \{\mathsf{False}, \mathsf{True}\}$, $\bot, \mathsf{W}, \top$ *are the smallest, neutral, biggest elements respectively, and* $\bot < \mathsf{False} < \mathsf{W} < \mathsf{True} < \top$; $H^+ = \{\mathsf{Very}, \mathsf{More}\}$ *and* $H^- = \{\mathsf{Possibly}, \mathsf{Less}\}$. *Consider the following clause set:*

1) $A(x)^{\mathsf{MF}} \vee B(z)^{\mathsf{MF}} \vee C(x)^{\mathsf{PT}}$
2) $C(y)^{\mathsf{MF}} \vee D(y)^{\mathsf{VMT}}$
3) $C(t)^{\mathsf{VVTrue}} \vee E(t, f(t))^{\mathsf{MF}}$
4) $E(a, u)^{\mathsf{T}}$
5) $D(a)^{\mathsf{MF}}$

where a *is a constant symbol and* x, y, z, t, u *are variables,* $\mathsf{T} = \mathsf{True}$, $\mathsf{F} = \mathsf{False}$, $\mathsf{V} = \mathsf{Very}$, $\mathsf{M} = \mathsf{More}$, $\mathsf{P} = \mathsf{Possibly}$, $\mathsf{L} = \mathsf{Less}$. *The reliability is defined based on the combination operator. At the beginning, the reliability of each clause is assigned to* T. *We have the following resolution inferences.*

$$\frac{(C(y)^{\mathsf{MF}} \vee D(y)^{\mathsf{VMT}}, \mathsf{T}) \qquad (C(t)^{\mathsf{VVT}} \vee E(t, f(t))^{\mathsf{MF}}, \mathsf{T})}{D(t)^{\mathsf{VMT}} \vee E(t, f(t))^{\mathsf{MF}}, \mathsf{MTrue})} [t/y]$$

$$\frac{D(t)^{\mathsf{VMT}} \vee E(t, f(t))^{\mathsf{MF}}, \mathsf{MT}) \qquad (D(a)^{\mathsf{MF}}, \mathsf{T})}{E(a, f(a))^{\mathsf{MF}}, \mathsf{MT})} [a/t]$$

$$\frac{E(a, f(a))^{\mathsf{MF}}, \mathsf{MT}) \qquad (E(a, u)^{\mathsf{True}}, \mathsf{T})}{(\Box, \mathsf{True})} [f(a)/u]$$

The empty clause is inferred, we conclude that the initial clause set is unsatisfiable and the reliability of the proof of unsatisfiability is True.

VI. Conclusion

We have presented a fuzzy first order linguistic logic having truth domain as a finite symmetrical refined hedge algebra. The syntax and semantic of logic have been defined. We have proposed a fuzzy linguistic resolution which is based on the usual resolution of two-valued logics. We have proved the soundness and completeness of the fuzzy linguistic resolution procedure. We have introduced the notion of reliability to capture the fact that a fuzzy linguistic resolution inference may involve literals with very contradictory truth values or with slightly contradictory truth values. In this respect we can estimate the reliability of a proof of a clause from a given set of clauses.

There are some lines of future works. It would be worth to investigate how to eliminate redundant resolution inferences to make the resolution procedure more efficient and effective in practice. Another interesting line of work is to consider other inference systems than resolution to widen the applicability scope of our work.

Acknowledgement

The authors gratefully acknowledge financial support from the Vietnam National Foundation for Science and Technology Development under Grant 102.04-2013.21.

References

[1] J. A. Robinson, "A machine-oriented logic based on the resolution principle," *J. ACM*, vol. 12, no. 1, pp. 23–41, 1965.
[2] R. C. T. Lee, "Fuzzy logic and the resolution principle," in *IJCAI*, 1971, pp. 560–567.
[3] M. M. Z. Shen, L. Ding, "Fuzzy resolution principle," in *Proc. 18th Internat. Symp. on Multiple-valued Logic*, 1989, pp. 210–215.
[4] D. Smutná and P. Vojtás, "Graded many-valued resolution with aggregation," *Fuzzy Sets and Systems*, vol. 143, no. 1, pp. 157–168, 2004.
[5] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.
[6] ——, "The concept of a linguistic variable and its application to approximate reasoning - i," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975.
[7] ——, "The concept of a linguistic variable and its application to approximate reasoning - ii," *Inf. Sci.*, vol. 8, no. 4, pp. 301–357, 1975.
[8] ——, "The concept of a linguistic variable and its application to approximate reasoning-iii," *Inf. Sci.*, vol. 9, no. 1, pp. 43–80, 1975.
[9] C. Nguyen and W. Wechler, *Hedge Algebras: An Algebraic Approach in Struture of Sets of Linguistic Truth Values*. Fuzzy Sets and Syst. 35, 1990, pp. 281–293.
[10] D.-K. Tran, V.-T. Vu, T.-V. Doan, and M.-T. Nguyen, "Fuzzy linguistic propositional logic based on refined hedge algebra," in *2013 IEEE International Conference on Fuzzy Systems*, 2013, pp. 1–8.
[11] P. H'ajek, *Metamathematics of Fuzzy Logic*. Dordrecht: Kluwer, 1998.
[12] V. Nov'ak, "A comprehensive theory of trichotomous evaluative linguistic expressions," *Fuzzy Sets and Systems*, vol. 159, no. 22, pp. 2939–2969, 2008.
[13] V. H. Le, F. Liu, and D. K. Tran, "Fuzzy linguistic logic programming and its applications," *TPLP*, vol. 9, no. 3, pp. 309–341, 2009.
[14] C.-H. Nguyen, D.-K. Tran, V.-N. Huynh, and H.-C. Nguyen, "Hedge algebras, linguistic-valued logic and their application to fuzzy reasoning," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 7, no. 4, pp. 347–361, 1999.
[15] A. P. Le and D. K. Tran, "A deductive method in linguistic reasoning," in *Uncertainty Reasoning and Knowledge Engineering (URKE), 2012 2nd International Conference on*, 2012, pp. 137–140.
[16] ——, "Linguistic reasoning based on generalized modus ponens with linguistic modifiers and hedge moving rules," in *Fuzzy Theory and it's Applications (iFUZZY), 2012 International Conference on*, 2012, pp. 82–86.
[17] C.-H. Nguyen and V.-N. Huynh, "An algebraic approach to linguistic hedges in zadeh's fuzzy logic," *Fuzzy Sets and Systems*, vol. 129, no. 2, pp. 229–254, 2002.