Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/110509

Linguistic-Valued Logics Based on Hedge Algebras and Applications to Approximate Reasoning

Nguyen Thi-Minh-Tam[1,*] *and Tran Duc-Khanh* [2]

[1] Faculty of Information Technology, Vinh University, Vietnam.
[2] Ho Chi Minh University of Technology (HUTECH), Vietnam.

Received: 14 Apr. 2017, Revised: 15 Aug. 2017, Accepted: 22 Aug. 2017
Published online: 1 Sep. 2017

Abstract: Humans reason by means of their own language and they can choose and decide alternatives by evaluating semantics of linguistic terms. The fundamental elements in human reasoning are sentences normally containing vague concepts, and these sentences have implicitly or explicitly a truth degree, which is often expressed also by linguistic terms such as *more or less false, very false, false, true, very true, more or less true, approximately true,* etc. In this article, we introduce a linguistic-valued predicate logic along with an inference system based on resolution. The truth domain of the logic is a refined hedge algebra, generated by a set of truth generators and a set of hedges. The syntax and semantics properties of the logic make sure that every formula has an equisatisfiable formula in conjunctive normal form, and a conjunctive normal form transformation algorithm can be devised. The resolution inference system, parameterized with a threshold of acceptability, is sound and complete. The linguistic-valued predicate logic together with the resolution inference system provides a framework for describing vague statements and mechanizing human reasoning in the presence of vagueness.

Keywords: Linguistic Truth, Hedge Algebra, Refined Hedge Algebra, Linguistic-valued Predicate Logic, Resolution, Approximate Reasoning

1 Introduction

Inference in logical languages is an approach to model human thinking and reasoning processes relying on accurate mathematical foundations. With a set of statements, called premises, one can infer new statements, called conclusions, so that if the premises are true then the conclusions must also be true.

The classical logic is suitable to reason with concepts that are crisp and well-defined in nature. However, in the real-world applications, uncertainty, which refers to a form of deficiency or imperfection in the information for which the truth of such information is not established definitely, is everywhere. Approximate reasoning methods, in general, are based on fuzzy sets theory and fuzzy logics [34]. The fuzzy logics deal with the vagueness in the knowledge, where a proposition is true only to some degree in the unit-interval $[0,1]$. For example, the statement "John is obese with degree 0.4?" indicates John is slightly obese. Here, the value 0.4 is the degree of membership that John is obese.

Furthermore, humans reason by means of their own language and they can choose and decide alternatives by evaluating semantics of linguistic terms. The fundamental elements in human reasoning are sentences normally containing vague concepts, and these sentences have implicitly or explicitly a truth degree, which is often expressed also by linguistic terms such as *more or less false, very false, false, true, very true, more or less true, approximately true,* etc. For example, one might want to find out to what degree a person, John, would have heart disease if the certainty that *"It is approximately true that an obese person would have heart disease"*, and *"It is very true that John is obese"*. Such knowledge cannot be expressed nor reasoned with standard logics or fuzzy logics. Therefore, in this paper, we are concerned with the two following issues:

1. constructing a *logical language* for vague statements in natural languages, and
2. designing an *inference system* for human reasoning with linguistic truth value.

* Corresponding author e-mail: nmtam@vinhuni.edu.vn

Linguistic-valued Predicate Logic

In 1965, Zadeh [34] introduced the fuzzy set theory and fuzzy logic to deal with reasoning under vagueness. And later Zadeh [28,29,30] used linguistic variables and linguistic values to formalize the reasoning in natural languages. Subsequently, Nguyen and Wechler [25] introduced the concept of hedge algebra which is an algebraic model for linguistic hedges in Zadeh's fuzzy logic.

According to Zadeh [28,29,30] the statements *"It is true that John is very smart"* and *"It is very true that John is smart"* are approximately equivalent. Therefore, instead of considering arbitrary linguistic variables, such as SMART, we can only consider a single linguistic variable TRUTH. And the logic under consideration can be viewed as a multiple-valued logic with the truth domain being the set of the linguistic values of the linguistic variable TRUTH.

As pointed out in [25] the set of linguistic values of the linguistic variable TRUTH, e.g., VeryTrue, MoreOrLessTrue, SlightlyTrue, VeryFalse, MoreOrLessFalse, SlightlyFalse,...} is a partially ordered set with an ordering meaning that an element describes a smaller degree of truth than another, and is generated from a set of generators, e.g., {True, False,...} and a set of operations also called hedges, e.g., {Very, Slightly, MoreOrLess,...}. This set can be represented by an abstract algebra with four components: the set of elements, the set of generators, the set of hedges, and the ordering. Furthermore, one can refine this abstract algebra into a De Morgan lattice by defining appropriately the join, meet, complementation operators [24,26,20].

Starting from these observations, we construct the Linguistic-valued Predicate Logic (LPL) which is expressive enough to describe vague statements in natural languages. LPL has the following features:

- –The truth domain is a refined hedge algebra generated by a set of truth generators and a set of hedges, for example {VeryTrue, SlightlyTrue, MoreOrLessTrue, VeryFalse, MoreOrLessFalse, SlightlyFalse,...}. It can be further refined to become a De Morgan lattice equipped with join, meet, complementation operators.
- –The syntax is similar to the syntax of two-valued predicate logics but augmented with truth operators. For example, the statement "John is very smart and works rather hard" is described by the formula

  $$(Smart(John)^{\mathsf{VeryTrue}} \wedge WorkHard(John)^{\mathsf{RatherTrue}})^{\mathsf{True}}$$

  where $Smart(John)$, $WorkHard(John)$ are atoms, and True, VeryTrue, RatherTrue are truth operators.
- –The semantics is defined so as to capture the meaning of vague statements. For instance, the meaning of the formula $Smart(John)^{\mathsf{SlightlyTrue}}$ is that the level of the vagueness of the claim *"John is smart"* is SlightlyTrue. That is, for a threshold of acceptability of SlightlyTrue the claim *"John is smart"* can be endorsed to be used in the reasoning process, but for a threshold of acceptability of True the claim *"John is smart"* is not sufficiently endorsed.

To summarize, our first contribution is the LPL for describing vague statements in natural languages. That provides a solution to Issue 1.

Approximate Reasoning

Since LPL is a multiple-valued fuzzy logic whose truth domain is the set of the linguistic values of the linguistic variable TRUTH, traditional automated reasoning methods based on the resolution for multiple-valued and fuzzy logics can be applied naturally. The resolution principle, initially designed for two-valued logics by Robinson [33], is the heart of many kinds of automated reasoning systems such as theorem proving and logic programming. The main advantage of resolution is that it is sound and complete for a wide range of multiple-valued and fuzzy logics [12,18,13,14,31,11,23,22,15].

Along the line of [31,23,15], we develop a resolution inference system for LPL. One of the main problems is that resolution usually works with a particular form of formulae called conjunctive normal form (CNF), but the LPL formulae may have nested truth operators, e.g.,

$$(Smart(John)^{\mathsf{VeryTrue}} \wedge WorkHard(John)^{\mathsf{RatherTrue}})^{\mathsf{True}}$$

and the traditional CNF transformation algorithm does not apply. We need a new CNF transformation algorithm. Fortunately, CNF transformation is feasible in LPL since its syntax and semantics are defined in such a way that every formula has an equisatisfiable CNF formula, and a CNF algorithm can be built by extending the traditional algorithm. The extension consists of driving truth operators inward the formulae. As an example, the formula

$$(Smart(John)^{\mathsf{VeryTrue}} \wedge WorkHard(John)^{\mathsf{RatherTrue}})^{\mathsf{False}}$$

can be converted into the CNF formula

$$Smart(John)^{\mathsf{VeryFalse}} \vee WorkHard(John)^{\mathsf{RatherFalse}}$$

by driving the truth operator False inward the formula

$$Smart(John)^{\mathsf{VeryTrue}} \wedge WorkHard(John)^{\mathsf{RatherTrue}}$$

On the other hand, our resolution procedure is parameterized with a threshold τ, called τ-resolution, so as to check whether a formula is τ-satisfiable, i.e., has the truth value greater or equal to τ under some interpretation. For example, if the threshold τ is VeryTrue then we have the following τ-resolution

$$\frac{WorkHard(John)^{\mathsf{T}} \vee Smart(John)^{\mathsf{VT}} \quad Smart(John)^{\mathsf{VVF}}}{WorkHard(John)^{\mathsf{T}}}$$

and $WorkHard(John)^{\mathsf{T}}$ is not τ-satisfiable.

(where $\mathsf{T}, \mathsf{VT}, \mathsf{VVF}$ are the abbreviations for $\mathsf{True}, \mathsf{VeryTrue}, \mathsf{VeryVeryFlase}$).

We prove the soundness and completeness of τ-resolution. Soundness means that if a τ-resolution derivation finds a τ-empty clause at some point, then the input formula is τ-unsatisfiable. Completeness means that if the input formula is τ-unsatisfiable then a τ-resolution derivation will find a τ-empty clause after a finite number of inferences. We show how τ-resolution can be adapted to other reasoning problems such as entailment checking and theorem proving.

The τ-resolution inferences are approximate by nature. Especially in the case where the threshold τ is low, the inferences are even more approximate, and we say that they have low confidence. To cope with this problem, we propose the so-called Δ-strategy for selecting inferences with the highest confidence. As the standard τ-resolution, the Δ-strategy is sound and complete.

In a nutshell, our second contribution is a resolution framework for mechanizing the human reasoning under vagueness. That constitutes a solution to Issue 2.

Structure of the paper The paper is organized as follows. Section 2 introduces hedge algebras. Section 3 defines LPL. Section 4 presents τ-resolution and proofs of its soundness and completeness. Section 5 shows how to apply τ-resolution in approximate reasoning. Section 6 discusses related work. Section 7 concludes and draws some future work.

2 Algebraic Approach to Linguistic Truth

Linguistic hedges were first studied by Lakoff [7]. Subsequently, Zadeh argued that the set of linguistic values of linguistic variables can be regarded as a formal language generated by a context-free grammar and hedges can be viewed as operators on fuzzy sets [28,29,30]. In [24,20,25] Nguyen et al. considered the sets of such linguistic values as *hedge algebras*. The class of hedge algebras enjoys interesting algebraic properties and is suitable to be used as truth domains for multiple-valued and fuzzy logics. Below we recall the concept of hedge algebras from [24,20,25] and show how it can be used in defining linguistic truth domains.

A linguistic variable is a variable whose values are words or sentences in some language, for example AGE is a linguistic variable if its values are linguistic, i.e., Old, Young, VeryOld, VeryYoung, etc ... A linguistic hedge is an operator which acts on the meaning of its operand, for example in the composite word VeryYoung the operator Very acts on the meaning of Young. Many linguistic variables have some primary vague concepts, for instance, the primary vague concepts of the linguistic variables AGE are Old and Young, of TRUTH are True and False, of HIGH are Tall and Small and so on... The primary vague concepts True, Old and Tall have "positive meaning" and the others have "negative meaning". We say that the hedge Very strengthens the positive or negative meanings, and the hedge Approximately weakens them. In this respect, the first characteristic of hedges is that: *every hedge either strengthens or weakens the positive or negative meaning of the primary vague concepts*. Besides, the hedge Very strengthens the meanings of the hedge Very, but weakens the meaning of the hedge Approximately, the hedge Approximately strengthens the meaning of the hedge Approximately but weakens the meanings of the hedges Very. This results in the second characteristic of hedges: *every hedge either strengthens or weakens the meanings of any other hedges*. On the other hand, observe that the element VeryVeryLessTrue expresses something about True but not about False, and likewise, VeryVeryLessFalse expresses something about False but not about True. Accordingly, the third characteristic of hedges is that: *all hedges have a local and separated meaning and the meaning of a term generated from a vague concept stems from the meaning of this vague concept.*

Let us now take a closer look at the linguistic variable TRUTH with the domain True, False, VeryTrue, VeryFalse, SlightlyTrue, SlightlyFalse, PossiblyTrue, PossiblyFalse, ApproximatelyTrue, VeryPossiblyTrue, ApproximatelyFalse,... This domain is a partially ordered set, and the ordering means that an element describes a smaller degree of truth than another. From an algebraic point of view, this set is generated from the basic elements True, False by using hedges Very, Possibly, Approximately, Slightly, ... and can be described by an abstract algebra with four components: the set of elements, the set of generators, the set of hedges, and the ordering over the set of elements. This abstract algebra is called a *hedge algebra*.

In the following, we consider a subclass of hedge algebras, called *refined hedge algebras* [20]. We first recall the formal definitions of refined hedge algebras. Then we show how a refined hedge algebra can be further refined to become a distributive lattice. This is done in two steps: first, we construct the distributive lattice of hedges by defining appropriate join and meet operators over the set of hedges; second, we construct the distributive lattice of linguistic truth values by defining appropriate join and meet operators over the set of linguistic truth values. Finally, we consider the subclass of *symmetric refined hedge algebras* where complementation operator can be defined over the set of linguistic truth values. In this way, a symmetric refined hedge algebra becomes a De Morgan lattice and can be readily used to define the truth domain for LPL. We assume the usual notions (such as posets, graded posets, lattice, modular lattice, finite lattice, free lattice, distributive lattice, etc...) from lattice theory in [5].

We consider a class of abstract algebras of the form $AX = (X, G, H, \leq)$, where X is the underlying set, G is the

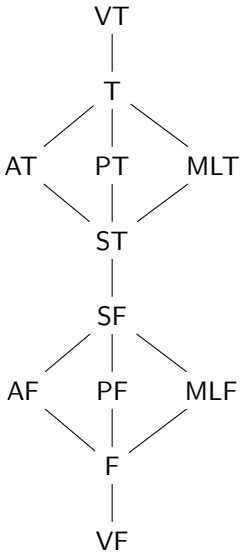


Fig. 1: A poset of values of the linguistic variable TRUTH

set of generators, H is a set of mappings from X to X, and $\leq$ is partial order on X. The set X is interpreted as the term set, G as a set of primary terms and special constants, H as a set of hedges, and $\leq$ as a semantic ordering relation. For convenience the image of $x \in X$ under $h \in H$ will be denoted by hx instead of $h(x)$.

Example 1.[20] Let X be a poset of values of the linguistic variable TRUTH as represented in Figure 1. Then X can be considered as an abstract algebra $AX = (X, G, H, \leq)$ such that

–$G = \{\mathsf{F}, \mathsf{T}\}$, where F, T stand for False, True, respectively;
–$H = \{\mathsf{V}, \mathsf{A}, \mathsf{P}, \mathsf{ML}, \mathsf{S}\}$, where V, A, P, ML, S stand for Very, Approximately, Possibly, MoreOrLess, Slightly, respectively;
–$\leq$ is a partially ordering relation over X as represented in Figure 1.

Definition 1.*Let $h, k \in H$, we say that:*

–*h and k are* converse *(or h is converse to k and vice versa) if $\forall x \in X(x \leq hx$ iff $x \geq kx)$;*
–*h and k are* compatible *if $\forall x \in X(x \leq hx$ iff $x \leq kx)$;*
–*h is* positive *with respect to k if $\forall x \in X$(either $kx \geq x$ implies $hkx \geq kx$ or $kx \leq x$ implies $hkx \leq kx$);*
–*h is* negative *with respect to k if $\forall x \in X$(either $kx \geq x$ implies $hkx \leq kx$ or $kx \leq x$ implies $hkx \geq kx$).*

We assume that:

(H1)each element $h \in H$ is an *ordering operation*, i.e. $\forall x \in X$(either $hx \geq x$ or $hx \leq x$);
(H2)the set H is decomposed into two non-empty disjoint subsets H^+, H^- such that for any $h \in H^+$ and $k \in H^-$, h and k are *converse*;

Fig. 2: Lattices of hedges $H^+ + I$

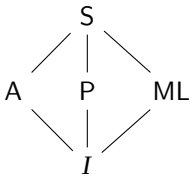


Fig. 3: Lattices of hedges $H^- + I$

(H3)the sets $H^+ + I$ and $H^- + I$, where $+$ stands for $\cup$, are finite lattices with unit-elements V and S, respectively, and zero-element I, where I is the identity element of H, i.e. $\forall x \in X(Ix = x)$. Since X, H^+ and H^- are disjoint, the partially ordering relation over each of X, H^+ and H^- will be denoted by the same notation $\leq$ without any confusion.

To simplify the notation, we use the superscript ^{c} to denote either the superscript $^+$ or $^-$.

*Example 2.*Let us consider the abstract algebra $AX = (X, G, H, \leq)$ in Example 1. Intuitively, we can see that:

–$H^+ = \{\mathsf{V}\}$; $H^- = \{\mathsf{A}, \mathsf{P}, \mathsf{ML}, \mathsf{S}\}$,
–$H^+ + I$ and $H^- + I$ are finite lattices;
–$\leq$ is a partially ordering relation over $H^+ + I$ and $H^- + I$ as represented in Figure 3.

It is clear that AX satisfies Assumption 2.

Definition 2.*We say that X and H are* semantically consistent *if the following conditions hold:*

1.*X is generated from the generators in G by means of hedges in H, i.e. elements of X are of the form $h_n \ldots h_1 a$ for $h_i \in H$, $i = 1, .., n$, and $a \in G$.*
2.*For any $h, k \in H^c + I$, where $c \in \{+, -\}$, $h < k$ in $H^c + I$ iff $\forall\, x \in X$((hx > x or kx > x implies hx < kx) and (hx < x or kx < x implies hx > kx)). And h, k are incomparable in $H^c + I$ iff $\forall\, x \in X$((hx $\neq$ x or kx $\neq$ x) implies hx and kx are incomparable).*

*Example 3.*Consider the abstract algebra $AX = (X, G, H, \leq)$ in Example 2. It is clear that X and H are semantically consistent.

In the following $H^c + I$ is a finite modular lattice, unless stated otherwise. According to [5] $H^c + I$ can be graded by its height function, and then decomposed into *graded classes* $H_i^c + I$, $i = 1, 2, \ldots, l$ where l is the length of the partially ordered set $H^c + I$. Additionally we assume that any two elements of $H^c + I$ belonging to different graded classes are comparable. Let LH_i^c be the free distributive

lattice generated by the set of incomparable elements of H_i^c, $\sqcup$ and $\sqcap$ be the corresponding *join* and *meet* operators. Let LH^c be the union of such H_i^c, and LH be the union of LH^+ and LH^-.

Proposition 1.*[20] $(LH^+ + I, \sqcup, \sqcap, I, V, \leq)$ and $(LH^- + I, \sqcup, \sqcap, I, S, \leq)$ are free distributive lattices with unit-elements V and S, respectively, and the zero-element I.*

*Example 4.*Consider the abstract algebra $AX = (X, G, H, \leq)$ in Example 2. Clearly, $H^c + I$ is a finite modular lattice, the graded classes are $\{\mathsf{S}\}, \{\mathsf{A}, \mathsf{P}, \mathsf{ML}\}, \{I\}, \{\mathsf{V}\}$. We can see that and any two elements of $H^c + I$ belonging to two different graded classes are comparable, in other words, there is linearly ordering relation between graded classes. Following the construction above, the obtained distributive lattices $LH^+ + I$ and $LH^- + I$ generated from $H^+ + I$ and $H^- + I$, respectively, are represented as in Figure 5, where

– $l_1 = \mathsf{A} \vee \mathsf{P} \vee \mathsf{ML}$;
– $l_2 = \mathsf{A} \vee \mathsf{P}$;
– $l_3 = \mathsf{A} \vee \mathsf{ML}$;
– $l_4 = \mathsf{P} \vee \mathsf{ML}$;
– $l_5 = (\mathsf{A} \vee \mathsf{P}) \wedge (\mathsf{A} \vee \mathsf{ML})$;
– $l_6 = (\mathsf{A} \vee \mathsf{P}) \wedge (\mathsf{P} \vee \mathsf{ML})$;
– $l_7 = (\mathsf{A} \vee \mathsf{ML}) \wedge (\mathsf{P} \vee \mathsf{ML})$;
– $l_8 = (\mathsf{A} \vee \mathsf{P}) \wedge (\mathsf{A} \vee \mathsf{ML}) \wedge (\mathsf{P} \vee \mathsf{ML})$
$= (\mathsf{A} \wedge \mathsf{P}) \vee (\mathsf{A} \wedge \mathsf{ML}) \vee (\mathsf{P} \wedge \mathsf{ML})$;
– $l_9 = (\mathsf{A} \wedge \mathsf{P}) \vee (\mathsf{A} \wedge \mathsf{ML})$;
– $l_{10} = (\mathsf{A} \wedge \mathsf{P}) \vee (\mathsf{P} \wedge \mathsf{ML})$;
– $l_{11} = (\mathsf{A} \wedge \mathsf{ML}) \vee (\mathsf{P} \wedge \mathsf{ML})$;
– $l_{12} = \mathsf{A} \wedge \mathsf{P}$; $l_{13} = \mathsf{A} \wedge \mathsf{ML}$; $l_{14} = \mathsf{P} \wedge \mathsf{ML}$;
– $l_{15} = \mathsf{A} \wedge \mathsf{P} \wedge \mathsf{ML}$.

Definition 3.*We say that H is* PN-homogenous *if V is positive (respectively negative) with respect to some operation in a graded class H_i^c then V is positive (respectively negative) with respect to any other ones in H_i^c, for any H_i^c.*

From now on, we denote H the set of primary hedges and LH the lattice of composed hedges constructed as above. For any element $x \in X$, $LH(x)$ denotes the set of all elements generated from x by means of hedges in LH. For any $X' \subset X$ and $LH' \subset LH$, $LH'(X')$ denotes the subset of X generated from the elements in X' by means of the operations in LH'. By $LH'[X']$ we denote the set $\{hx : h \in LH' \text{ and } x \in X'\}$. By SI^c we denote the set of all indexes i which are not single-class elements. LH_i^c denotes the graded class i of LH^c, LH^* denotes the set of all strings of hedges in LH. Let us denote by USO the set of two unit-elements V and S of $LH^+ + I$ and $LH^- + I$.

Definition 4.*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^+.

Fig. 4: Distributive lattice $LH^+ + I$

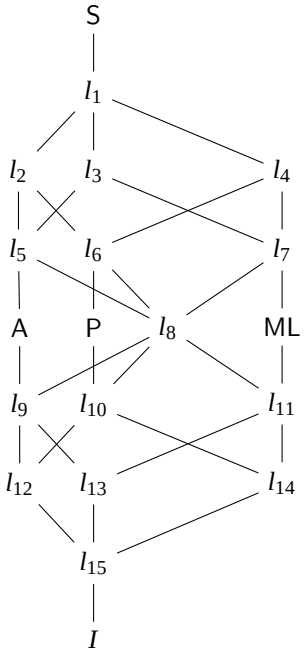


Fig. 5: Distributive lattice $LH^- + I$

2.The unit operation V of $LH^+ + I$ is either positive or negative w.r.t. any operation in LH. In addition, LH satisfies 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 so are for any 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

(a)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 $\forall z \in LH(hx)$ such that $z \not\leq \delta hx$.

(b)If $\{h, k\} \not\subset LH_i^c$, then $h'hx \leq k'kx$, for every $h', k' \in USO$.

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 there is $v \in LH(hx)$, 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 $h' \in UOS$.

Example 5.[20] Consider the abstract algebra $AX = (X_1, G, LH, \leq)$ such that, as considered in Examples 1 and 2 , $G = \{\mathsf{F}, \mathsf{T}\}$, $H = \{\mathsf{V}, \mathsf{A}, \mathsf{P}, \mathsf{ML}, \mathsf{S}\}$, but $X_1 = \{ha : h \in LH + I, a \in G\}$ which is ordered as in Figure 6, where $L(\mathsf{A}, \mathsf{P}, \mathsf{ML})$ denotes the lattice generated from the incomparable A, P, ML, and $L(\mathsf{A}, \mathsf{P}, \mathsf{ML})[a]$ denotes the set $\{ha : h \in L(A, P, ML)\}$. Here hx is defined as follows: for every hedge $h \in LH$, $h\mathsf{T}$ and $h\mathsf{F}$ are defined as elements given in Figure 6 and $hx = x$ for $x \neq \mathsf{F}$ and $x \neq \mathsf{T}$. We can see that AX satisfies Definition 4.

Fig. 6: The poset of Example 5

It is possible to define the *join* and *meet* operators over AX, denoted with $\cup$ and $\cap$, respectively. They are determined recursively based on the corresponding join and meet operators over the hedge lattice LH. Let x and y be two incomparable elements of AX, we can represent them in the form $x = \delta hu$ and $y = \gamma ku$, where $h, k \in LH_i^c$ for some $i \in SI^c$ and $\delta, \gamma \in LH^*$. Then:

– $x \cup y = \delta hu \cup \gamma ku = \delta u' \cup \gamma u'$
– $x \cap y = \delta hu \cap \gamma ku = \delta u'' \cup \gamma u''$

where $u' = (h \sqcup k)u$ and $u'' = (h \sqcap k)u$ if $hu > u$; $u' = (h \sqcap k)u$ and $u'' = (h \sqcup k)u$ if $hu < u$.

Proposition 2.*[20] Let $AX = (X, G, LH, \leq)$ be a refined hedge algebra. Then $(X, G, LH, \leq, \cup, \cap)$ is a distributive lattice if G is linearly ordered.*

An element p of G is a *primary generator* if $hp \neq p$ for all $h \in LH$. An element g of G is a *positive generator* if $Vg \geq g$, and is a *negative generator* if $Vg \leq g$. An element f of G is a fixed point if $hf = f$ for all $h \in LH$.
assumption We assume that $AX = (X, G, LH, \leq)$ is a refined hedge algebra such that:

– $G = \{\bot, c^-, \mathsf{W}, c^+, \top\}$,
– the elements c^- and c^+ are primary negative and positive generators, respectively,
– the elements $\bot$, W and $\top$ are fixed points, called the least, the neutral and the greatest elements in X, respectively.

An expression $h_n \ldots h_1 y$ is a *canonical representation* of x with respect to y in AX if $x = h_n \ldots h_1 y$ and $h_i \ldots h_1 y \neq h_{i-1} \ldots h_1 y$ for every $i \leq n$. Let x be an element of AX and the canonical representation of x is $x = h_n ... h_1 a$ where $a \in \{c^-, c^+\}$. The *complementary element* of x is an element y such that $y = h_n ... h_1 a'$ where $a' \in \{c^-, c^+\}$ and $a' \neq a$. The complementary element of $\top$ is $\bot$, and vice versa. The complementary element of W is itself.

Definition 5.*Let $AX = (X, G, LH, \leq)$ be a refined hedge algebra satisfying Assumption 2. Then AX is said to be a* symmetric refined hedge algebra *if every element in X has a unique complementary element in X.*

Let x and y be two elements of the symmetric refined hedge algebra AX, then:

– the *concept negation* $-$ is an unary operator defined such as $-x$ is the complementary element of x;
– the *concept implication* $\Rightarrow$ is a binary operator defined as: $x \Rightarrow y = -x \cup y$.

In practice, we only have to deal with linguistic truth values containing a limited number of hedges. Therefore, it is reasonable to assume that the set of elements of a symmetric refined hedge algebra is finite.
Assumption
We assume a symmetric refined hedge algebra

$$AX = (LH_p[G], G, LH, \leq)$$

where:

$$LH_p[G] = \{h_n ... h_1 a : h_i \in LH + I, a \in G, n \leq p\}$$

Since the operations $\cup$, $\cap$, $-$, $\Rightarrow$ can be defined for the symmetric refined hedge algebra $AX = (LH_p[G], G, LH, \leq)$ satisfying Assumptions 2, 2 and 2, we can write:

$$AX = (LH_p[G], G, LH, \leq, \cup, \cap, -, \Rightarrow)$$

Theorem 1.*[20] $AX = (LH_p[G], G, LH, \leq, \cup, \cap, -, \Rightarrow)$ is a complete distributive lattice. Furthermore for all $x,y \in LH_p[G]$, for all $h \in LH$, we have:*

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

Definition 6.*Let λ_1 and λ_2 be elements of $LH_p(G)$. The operation $\otimes$ is defined as follows*

$$\lambda_1 \otimes \lambda_2 = (\lambda_1 \cap \lambda_2) \cup (-\lambda_1 \cap -\lambda_2)$$

Definition 7.*A* linguistic truth domain *is a symmetric refined hedge algebra $AX = (LH_p[G], G, LH, \leq, \cup, \cap, -, \Rightarrow)$, where*

- $G = \{\bot, \mathsf{False}, \mathsf{W}, \mathsf{True}, \top\}$;
- $\bot < \mathsf{False} < \mathsf{W} < \mathsf{True} < \top$;
- False, True *are the negative and positive primary generator, respectively;*
- $\bot, \mathsf{W}, \top$ *are the smallest, neutral, biggest elements, respectively.*

3 Linguistic-valued Predicate Logic

In this section, we define the syntax and semantics of LPL. The syntax and semantics properties make sure that every formula has an equisatisfiable formula in CNF. We give an algorithm for transforming an arbitrary formula into CNF. We also show that the Herbrand theorem for two-valued predicate logics can be smoothly adapted to LPL.

3.1 Syntax and Semantics

Definition 8.*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 ($n \geq 0$);*
- *function symbols 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 symbols with 0-arity is called a logical constant symbol;*
- *logical connectives: $\neg, \vee, \wedge, \rightarrow$;*
- *quantifiers: $\forall, \exists$;*
- *auxiliary symbols: $\square, (,), \ldots$;*

Definition 9.*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 10.*An atomic formula is an expression of the form $(P(t_1, \ldots, t_n))^{\lambda}$, where P is an n-ary predicate symbol of the alphabet, λ is an element of $LH_p(G)$ and $t_1, \ldots, t_n$ are terms.*

Definition 11.*Formulae are defined recursively as follows:*

- *any atomic formula is a formula,*
- *if ϕ, ψ are formulae, then $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$ and $\phi \rightarrow \psi$ are formulae,*
- *if x is a variable and ϕ is a formula then $\forall x\phi, \exists x\phi$ are formulae,*
- *if ϕ is a formula and λ is an element of $LH_p(G)$ then ϕ^{λ} is a formula, ϕ^{λ} is also called a literal if ϕ is an atom.*

A clause is a finite set of literals, is usually written as a disjunction $l_1 \vee l_2 \vee \ldots \vee l_n$, where l_i is a literal (for $i = 1, \ldots, n$). A formula is said to be in a conjunctive normal form (CNF) if it is a conjunction of clauses. 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 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: $\{t_1/v_1, t_2/v_2, .., t_n/v_n\}$, where $v_i (i = 1..n)$ are dinstinct.

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 a substitution ϕ such that $\sigma = \gamma \circ \phi$. We say that γ is the most general unifier (m.g.u for short) if there is no unifier more general than γ.

Definition 12.*A structure* M *of an alphabet $\mathscr{A}$ is a triplet (D, I^f, I^p), where:*

–*D is a nonempty set called the domain of the structure, and:*
  –$\mathscr{F}_D$ *is the set of functions on D:* $\mathscr{F}_D = \{f_D | f_D : D^n \longmapsto D, n > 0\}$
  –$\mathscr{P}_D$ *is the set of relations on D:* $\mathscr{P}_D = \{P_D | P_D : D^n \longmapsto LH_p(G), n \geq 0\}$
–$I^f : FS \longmapsto \mathscr{F}_D$, *for every n-ary function symbol* f, f *is assigned to an element in* $\mathscr{F}_D$
–$I^p : PS \longmapsto \mathscr{P}_D$, *for every n-ary predicate symbol P, P is assigned to an element in* $\mathscr{P}_D$

Definition 13.*An assignment* σ *is a mapping of the set of variables* V *into the domain* D, $\sigma : V \longmapsto D$.

Definition 14.*An interpretation* $\mathscr{I}$ *is a pair* (M, σ), *where* M *is a structure and* σ *is an assignment.*

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

Definition 15.*The evaluation of a term under an interpretation* $\mathscr{I} = (\mathsf{M}, \sigma)$ *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 16.*The evaluation of formulae under an interpretation* $\mathscr{I} = (\mathsf{M}, \sigma)$ *is determined recursively as follows:*

–$[|\phi^{\lambda}|]^{\mathsf{M}}_{\sigma} = \lambda \otimes [|\phi|]^{\mathsf{M}}_{\sigma}$
–$[|\neg\phi|]^{\mathsf{M}}_{\sigma} = -[|\phi|]^{\mathsf{M}}_{\sigma}$
–$[|\phi \wedge \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \cap [|\varphi|]^{\mathsf{M}}_{\sigma}$
–$[|\phi \vee \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \cup [|\varphi|]^{\mathsf{M}}_{\sigma}$
–$[|\phi \rightarrow \varphi|]^{\mathsf{M}}_{\sigma} = [|\phi|]^{\mathsf{M}}_{\sigma} \Rightarrow [|\varphi|]^{\mathsf{M}}_{\sigma}$
–$[|\exists x\phi|]^{M}_{\sigma} = \bigcup_{c \in D}\{[|\phi|]^{M}_{\sigma'} | \sigma' = \sigma \cup \{x \longmapsto c\}\}$
–$[|\forall x\phi|]^{M}_{\sigma} = \bigcap_{c \in D}\{[|\phi|]^{M}_{\sigma'} | \sigma' = \sigma \cup \{x \longmapsto c\}\}$

Definition 17.*Let* τ *be an element of* $LH_p(G)$, $\mathscr{I} = (\mathsf{M}, \sigma)$ *be an interpretation, and* ϕ *be a formula. Then we say that*

–$\mathscr{I}$ τ*-satisfies* ϕ, *or* $\mathscr{I}$ *is a* τ*-model of* ϕ, *iff* $[|\phi|]^{\mathsf{M}}_{\sigma} \geq \tau$, *denoted by* $\mathscr{I} \models_{\tau} \phi$;
–ϕ *is* τ*-satisfiable iff it has a* τ*-model;*
–ϕ *is* τ*-unsatisfiable iff it has no* τ*-model;*
–ϕ *is* τ*-valid in* M *if* $[|\phi|]^{\mathsf{M}}_{\sigma} \geq \tau$ *for all assignments* σ, *denoted by* $\mathsf{M} \models_{\tau} \phi$. M *is called a* τ*-model of* ϕ;
–ϕ *is a* τ*-tautology iff it is* τ*-valid in all structures* M, *denoted by* $\models_{\tau} \phi$.
–φ *is a* τ*-consequence of* ϕ, *or* ϕ τ*-entails* φ, *denoted by* $\phi \models_{\tau} \varphi$, *iff for every interpretation* $\mathscr{I}$, $\mathscr{I} \models_{\tau} \phi$ *implies that* $\mathscr{I} \models_{\tau} \varphi$.
–ϕ *and* φ *are said to be* τ*-equisatisfiable if* ϕ *is* τ*-satisfiable iff* φ *is* τ*-satisfiable.*

3.2 Conjunctive Normal Form

Below we give a rule-based algorithm to transform an arbitrary formula into an equisatisfiable CNF formula. The algorithm consists of the following steps.

1.Eliminate implication:
  –$\phi \rightarrow \varphi \Longrightarrow_{CNF} \neg\phi \vee \varphi$
2.Move negation inward:
  –$\neg\neg\phi \Longrightarrow_{CNF} \phi$
  –$\neg(\phi^{\lambda}) \Longrightarrow_{CNF} \phi^{-\lambda}$
  –$\neg(\phi \vee \varphi) \Longrightarrow_{CNF} \neg\phi \wedge \neg\varphi$
  –$\neg(\phi \wedge \varphi) \Longrightarrow_{CNF} \neg\phi \vee \neg\varphi$
  –$\neg(\exists x\phi) \Longrightarrow_{CNF} \forall x\neg\phi$
  –$\neg(\forall x\phi) \Longrightarrow_{CNF} \exists x\neg\phi$
3.Move operators inward:
  –$(\phi^{\lambda_1})^{\lambda_2} \Longrightarrow_{CNF} \phi^{\lambda_1 \otimes \lambda_2}$
  –$(\neg\phi)^{\lambda} \Longrightarrow_{CNF} \phi^{-\lambda}$
  –$(\phi \vee \varphi)^{\lambda} \Longrightarrow_{CNF} \phi^{\lambda} \vee \varphi^{\lambda}$ if $\lambda \geq \mathsf{W}$
  –$(\phi \wedge \varphi)^{\lambda} \Longrightarrow_{CNF} \phi^{\lambda} \wedge \varphi^{\lambda}$ if $\lambda \geq \mathsf{W}$
  –$(\phi \vee \varphi)^{\lambda} \Longrightarrow_{CNF} \phi^{\lambda} \wedge \varphi^{\lambda}$ if $\lambda < \mathsf{W}$
  –$(\phi \wedge \varphi)^{\lambda} \Longrightarrow_{CNF} \phi^{\lambda} \vee \varphi^{\lambda}$ if $\lambda < \mathsf{W}$
  –$(\forall x\phi)^{\lambda} \Longrightarrow_{CNF} \forall x\phi^{\lambda}$ if $\lambda \geq \mathsf{W}$
  –$(\exists x\phi)^{\lambda} \Longrightarrow_{CNF} \exists x\phi^{\lambda}$ if $\lambda \geq \mathsf{W}$
  –$(\exists x\phi)^{\lambda} \Longrightarrow_{CNF} \forall x\phi^{\lambda}$ if $\lambda < \mathsf{W}$
  –$(\forall x\phi)^{\lambda} \Longrightarrow_{CNF} \exists x\phi^{\lambda}$ if $\lambda < \mathsf{W}$
4.Standardize variables:
  –If two variables have the same name but are in two different clauses then rename one of them.
5.Move quantifiers outward:
  –$(Qx\phi) \wedge \varphi \Longrightarrow_{CNF} Qx(\phi \wedge \varphi)$ (x not free in φ)
  –$(Qx\phi) \vee \varphi \Longrightarrow_{CNF} Qx(\phi \vee \varphi)$ (x not free in φ)
  –$\phi \wedge (Qx\varphi) \Longrightarrow_{CNF} Qx(\phi \wedge \varphi)$ (x not free in ϕ)
  –$\phi \vee (Qx\varphi) \Longrightarrow_{CNF} Qx(\phi \vee \varphi)$ (x not free in ϕ)
  where $Q \in \{\exists, \forall\}$
6.Eliminate existential quantifiers:
  - $\forall x_1 \ldots \forall x_n \exists x\phi \Longrightarrow_{CNF} \forall x_1 \ldots \forall x_n \phi[x := \pi_{(x_1,\ldots,x_n)}]$
  where π is a new n-ary function symbol, also called "Skolem function"
7.Eliminate universal quantifiers:
  –$\forall x\phi \Longrightarrow_{CNF} \phi$
8.Distribute disjunctions inward over conjunctions:
  –$\phi \vee (\varphi \wedge \gamma) \Longrightarrow_{CNF} (\phi \vee \varphi) \wedge (\phi \vee \gamma)$
9.Eliminate duplicates
  –$\phi \vee \varphi \vee \varphi \Longrightarrow_{CNF} \phi \vee \varphi$
  –$\phi \wedge \varphi \wedge \varphi \Longrightarrow_{CNF} \phi \wedge \varphi$

To prove that $\Longrightarrow_{CNF}$ preserves τ-satisfiability we need the following results.

Proposition 3.$\otimes$ *is associative and commutative.*

*Proof.*We have $\lambda_1 \otimes \lambda_2 = (\lambda_1 \cap \lambda_2) \cup (-\lambda_1 \cap -\lambda_2) = (\lambda_2 \cap \lambda_1) \cup (-\lambda_2 \cap -\lambda_1)$. This means $\otimes$ is commutative.
By definition
$(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = (\lambda_1 \cap \lambda_2 \cap \lambda_3) \cup (-\lambda_1 \cap -\lambda_2 \cap \lambda_3) \cup (\lambda_1 \cap -\lambda_1 \cap -\lambda_3) \cup (-\lambda_1 \cap \lambda_2 \cap -\lambda_3) \cup (\lambda_1 \cap -\lambda_2 \cap -\lambda_3) \cup (\lambda_2 \cap -\lambda_2 \cap -\lambda_3)$

and
$\lambda_1 \otimes (\lambda_2 \otimes \lambda_3) = (\lambda_1 \cap \lambda_2 \cap \lambda_3) \cup (\lambda_1 \cap -\lambda_2 \cap -\lambda_3) \cup (-\lambda_1 \cap \lambda_2 \cap -\lambda_2) \cup (-\lambda_1 \cap -\lambda_2 \cap \lambda_3) \cup (-\lambda_1 \cap \lambda_2 \cap -\lambda_3) \cup (-\lambda_1 \cap \lambda_3 \cap -\lambda_3)$.
Consider the following cases:

– $\mathsf{W} \leq \lambda_1, \lambda_2, \lambda_3$: then $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = \lambda_1 \cap \lambda_2 \cap \lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$;
– $\lambda_1, \lambda_2, \lambda_3 < \mathsf{W}$: then $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = \lambda_1 \cup \lambda_2 \cup \lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$;
– $\lambda_2, \lambda_3 < \mathsf{W} \leq \lambda_1$: then $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = \lambda_1 \cap -\lambda_2 \cap -\lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$;
– $\lambda_3 < \mathsf{W} \leq \lambda_1, \lambda_2$: then $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = -\lambda_1 \cup -\lambda_2 \cup \lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$.

In all cases we have $(\lambda_1 \otimes \lambda_2) \otimes \lambda_3 = \lambda_1 \otimes (\lambda_2 \otimes \lambda_3)$. This means $\otimes$ is associative.

Proposition 4. $-(\lambda_1 \otimes \lambda_2) = -\lambda_1 \otimes \lambda_2 = \lambda_1 \otimes -\lambda_2$

Proof. We have that

$$-\lambda_1 \otimes \lambda_2 = (\lambda_1 \cap -\lambda_2) \cup (-\lambda_1 \cap \lambda_2) = \lambda_1 \otimes -\lambda_2$$

and

$$-(\lambda_1 \otimes \lambda_2) = (\lambda_1 \cap -\lambda_1) \cup (\lambda_1 \cap -\lambda_2) \cup (-\lambda_1 \cap \lambda_2) \cup (\lambda_2 \cap -\lambda_2)$$

Consider the following cases:

– $\lambda_1, \lambda_2 \geq \mathsf{W}$
Then $-(\lambda_1 \otimes \lambda_2) = -\lambda_1 \cup -\lambda_2 = -\lambda_1 \otimes \lambda_2 = \lambda_1 \otimes -\lambda_2$
– $\mathsf{W} > \lambda_1, \lambda_2$
Then $-(\lambda_1 \otimes \lambda_2) = \lambda_1 \cup \lambda_2 = -\lambda_1 \otimes \lambda_2 = \lambda_1 \otimes -\lambda_2$
– $\lambda_1 \geq \mathsf{W} > \lambda_2$
Then $-(\lambda_1 \otimes \lambda_2) = \lambda_1 \cap -\lambda_2 = -\lambda_1 \otimes \lambda_2 = \lambda_1 \otimes -\lambda_2$

In all cases we have $-(\lambda_1 \otimes \lambda_2) = -\lambda_1 \otimes \lambda_2 = \lambda_1 \otimes -\lambda_2$.

Proposition 5. *Let $\mathscr{I} = (\mathsf{M}, \sigma)$ be an interpretation, and ϕ, φ, ψ be formulae. Then*

– $[|(\phi^{\lambda_1})^{\lambda_2}|]^{\mathsf{M}}_{\sigma} = [|\phi^{\lambda_1 \otimes \lambda_2}|]^{\mathsf{M}}_{\sigma}$
– $[|\neg(\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$
– $[|(\neg\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$
– $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda \geq \mathsf{W}$
– $[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda \geq \mathsf{W}$
– $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda < \mathsf{W}$
– $[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda < \mathsf{W}$
– $[|(\forall x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\forall x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda \geq \mathsf{W}$
– $[|(\exists x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\exists x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda \geq \mathsf{W}$
– $[|(\forall x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\exists x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda < \mathsf{W}$
– $[|(\exists x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\forall x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ *if* $\lambda < \mathsf{W}$

Proof. The first property follows from Proposition 3.

– $[|(\phi^{\lambda_1})^{\lambda_2}|]^{\mathsf{M}}_{\sigma} = [|\phi^{\lambda_1 \otimes \lambda_2}|]^{\mathsf{M}}_{\sigma}$
Let $[|\phi|]^{\mathsf{M}}_{\sigma} = \lambda$. Then we have $[|(\phi^{\lambda_1})^{\lambda_2}|]^{\mathsf{M}}_{\sigma} = (\lambda \otimes \lambda_1) \otimes \lambda_2 = \lambda \otimes (\lambda_1 \otimes \lambda_2) = [|\phi^{\lambda_1 \otimes \lambda_2}|]^{\mathsf{M}}_{\sigma}$.

The next two properties follow from Proposition 4.

– $[|\neg(\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$
Let $[|\phi|]^{\mathsf{M}}_{\sigma} = \lambda'$. Then we have $[|\neg(\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = -\lambda \otimes \lambda' = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$.
– $[|(\neg\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$
Let $[|\phi|]^{\mathsf{M}}_{\sigma} = \lambda'$. Then we have $[|(\neg\phi^{\lambda})|]^{\mathsf{M}}_{\sigma} = \lambda \otimes -\lambda' = -\lambda \otimes \lambda' = [|\phi^{-\lambda}|]^{\mathsf{M}}_{\sigma}$.

We now prove the following properties.

– $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ if $\lambda \geq \mathsf{W}$
Let $[|\phi|]^{\mathsf{M}}_{\sigma} = \lambda_1$ and $[|\varphi|]^{\mathsf{M}}_{\sigma} = \lambda_2$. Then we have $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = \lambda \otimes (\lambda_1 \cup \lambda_2) = (\lambda \cap (\lambda_1 \cup \lambda_2)) \cup (-\lambda \cap -(\lambda_1 \cup \lambda_2)) = (\lambda \cap \lambda_1) \cup (\lambda \cap \lambda_2) \cup (-\lambda \cap -\lambda_1 \cap -\lambda_2)$
and
$[|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma} = (\lambda \otimes \lambda_1) \cup (\lambda \otimes \lambda_2) = (\lambda \cap \lambda_1) \cup (\lambda \cap \lambda_2) \cup (-\lambda \cap -\lambda_1) \cup (-\lambda \cap -\lambda_2)$.
We consider the following cases

– $\lambda \geq \mathsf{W}$ and $\lambda_1, \lambda_2 \geq \mathsf{W}$
Then we have $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = (\lambda \cap \lambda_1) \cup (\lambda \cap \lambda_2) = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.
– $\lambda \geq \mathsf{W}$ and $\mathsf{W} > \lambda_1, \lambda_2$
Then we have $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = -\lambda \cup \lambda_1 \cup \lambda_2 = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.
– $\lambda \geq \mathsf{W}$ and $\lambda_1 \geq \mathsf{W} > \lambda_2$
Then we have $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = \lambda \cap \lambda_1 = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.

– $[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ if $\lambda \geq \mathsf{W}$
Let $[|\phi|]^{\mathsf{M}}_{\sigma} = \lambda_1$ and $[|\varphi|]^{\mathsf{M}}_{\sigma} = \lambda_2$. Then we have $[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = \lambda \otimes (\lambda_1 \cap \lambda_2) = (\lambda \cap (\lambda_1 \cap \lambda_2)) \cup (-\lambda \cap -(\lambda_1 \cap \lambda_2)) = (\lambda \cap \lambda_1 \cap \lambda_2) \cup (-\lambda \cap -\lambda_1) \cup (-\lambda \cap -\lambda_2)$
and
$[|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma} = (\lambda \otimes \lambda_1) \cap (\lambda \otimes \lambda_2) = (\lambda \cap \lambda_1 \cap \lambda_2) \cup (\lambda \cap -\lambda \cap \lambda_1 \cap -\lambda_2) \cup (\lambda \cap -\lambda \cap -\lambda_1 \cap \lambda_2) \cup (-\lambda \cap -\lambda_1 \cap -\lambda_2)$.
We consider the following cases

– $\lambda \geq \mathsf{W}$ and $\lambda_1, \lambda_2 \geq \mathsf{W}$
Then we have
$[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = \lambda \cap \lambda_1 \cap \lambda_2 = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.
– $\lambda \geq \mathsf{W}$ and $\mathsf{W} > \lambda_1, \lambda_2$
Then we have
$[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = -\lambda \cup \lambda_1 \cap \lambda_2 = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.
– $\lambda \geq \mathsf{W}$ and $\lambda_1 \geq \mathsf{W} > \lambda_2$
Then we have
$[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = -\lambda \cup \lambda_2 = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$.

The proof for the following properties is similar to the previous ones.

– $[|(\phi \vee \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \wedge (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ if $\lambda < \mathsf{W}$
– $[|(\phi \wedge \varphi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|(\phi^{\lambda}) \vee (\varphi^{\lambda})|]^{\mathsf{M}}_{\sigma}$ if $\lambda < \mathsf{W}$

Now let us prove the following properties.

– $[|(\forall x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\forall x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ if $\lambda \geq \mathsf{W}$

We have

$$\begin{aligned}[|(\forall x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} &= \lambda \otimes [|\forall x\phi|]^{\mathsf{M}}_{\sigma}\\ &= \lambda \otimes \bigcap_{c\in D}\{[|\phi|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= \bigcap_{c\in D}\{\lambda \otimes [|\phi|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= \bigcap_{c\in D}\{[|\phi^{\lambda}|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= [|\forall x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}.\end{aligned}$$

– $[|(\exists x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\exists x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ if $\lambda \geq \mathsf{W}$

We have

$$\begin{aligned}[|(\exists x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} &= \lambda \otimes [|\exists x\phi|]^{\mathsf{M}}_{\sigma}\\ &= \lambda \otimes \bigcup_{c\in D}\{[|\phi|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= \bigcup_{c\in D}\{\lambda \otimes [|\phi|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= \bigcup_{c\in D}\{[|\phi^{\lambda}|]^{\mathsf{M}}_{\sigma'} | \sigma' = \sigma \cup \{x \mapsto c\}\}\\ &= [|\exists x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}.\end{aligned}$$

The proof for the following properties is similar the two previous ones.

– $[|(\forall x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\exists x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ if $\lambda < \mathsf{W}$
– $[|(\exists x\phi)^{\lambda}|]^{\mathsf{M}}_{\sigma} = [|\forall x\phi^{\lambda}|]^{\mathsf{M}}_{\sigma}$ if $\lambda < \mathsf{W}$

We are now ready to prove that $\Longrightarrow_{CNF}$ preserves τ-satisfiability.

Theorem 2.*Let ϕ and φ be formulae such that $\phi \Longrightarrow_{CNF} \varphi$. Then ϕ and φ are τ-equisatisfiable.*

*Proof.*Steps 1-3 and 8-9 preserve τ-satisfiability because of Theorem 1, Propositions 4 and 5. Step 4 clearly preserves τ-satisfiability. Steps 5-7 preserve τ-satisfiability by definition of the quantifiers $\forall$ and $\exists$.

3.3 Herbrand Theorem

The Herbrand universe, or H-universe for short, of an alphabet $\mathscr{A}$, denoted by $U(\mathscr{A})$, is the set of all ground terms built over $\mathscr{A}$. The set of all ground atoms built over $\mathscr{A}$ is called the Herbrand base, or H-base for short, denoted by $B(\mathscr{A})$. A Herbrand structure, or H-structure for short, of an alphabet $\mathscr{A}$ is a structure having the domain which is the Herbrand universe $U(\mathscr{A})$. A Herbrand interpretation, or H-interpretation for short, of an alphabet $\mathscr{A}$ is a pair (M_H, σ_H), where M_H is an H-structure of the alphabet $\mathscr{A}$ and $\sigma_H : V \longrightarrow U(\mathscr{A})$ 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 $\mathscr{A}(S)$ 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 $\mathscr{A}(S)$. The H-base of S, denoted by $B(S)$, is the H-base of the alphabet $\mathscr{A}(S)$. An H-structure of S is an H-structure of the alphabet $\mathscr{A}(S)$. An H-interpretation of S is an H-interpretation of the alphabet $\mathscr{A}(S)$.

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

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

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

Then, we have:

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

This shows that if $\mathscr{I}$ τ-satisfies S then $\mathscr{H}$ τ-satisfies S as well.

($\Leftarrow$) If an H-interpretation $\mathscr{H}$ τ-satisfies S, then it is obvious that S is τ-satisfiable.

Let S be a set of ground clauses and $B(S)$ be the H-base of S. For $\tau > \mathsf{W}$, a τ-semantic tree of S is an 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 labeled with $A_i \leq -\tau$, and the right edge of each node at the level i is labeled with $A_i \geq \tau$ (cf. Fig 7).
– 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 appear in any other levels.

There are different τ-semantic trees for a given set of clauses, depending on the choice of atoms at each level of the tree.

Notice that each path from the root to a certain leaf in a τ-semantic tree corresponds to an H-interpretation $\mathscr{I}$ of the clause set S, such that $\mathscr{I}(A_i) \leq -\tau$ or $\mathscr{I}(A_i) \geq \tau$, for $i = 1,\ldots,n$. We do not consider the H-interpretations $\mathscr{I}$ such that $-\tau < \mathscr{I}(A_i) < \tau$ because they will not τ-satisfy $A_i^{\lambda_i}$ for every $\lambda_i \in LH_p[G]$, for $i = 1,\ldots,n$.

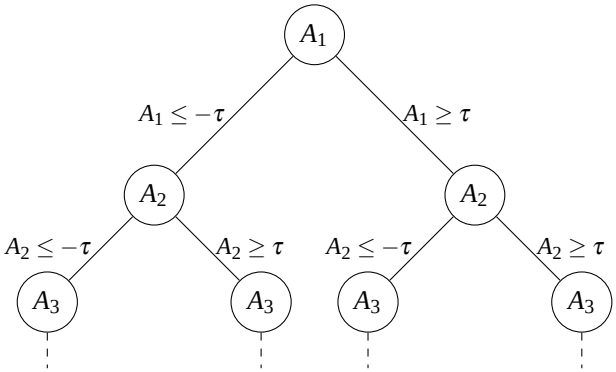


Fig. 7: τ-semantic tree

Let T be a τ-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 $\mathscr{I}$ corresponding to a branch of T containing N such that C is not τ-satisfied by $\mathscr{I}$. 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 1.*There always exists an inference node on the finite closed tree.*

*Proof.*Assume that we have a closed tree T . Because T has a 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 the parent node of j. By definition of the closed tree, i cannot be failure node. This implies that 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.

Lemma 2.*Let S be a set of the ground clause. Then S is τ-unsatisfiable iff for every τ-semantic tree of S, there exists a finite closed tree.*

Proof.($\Rightarrow$) Suppose that S is τ-unsatisfiable and T is a τ-semantic tree of S. Let B be a branch of T, we denote $\mathscr{I}_B$ the H-interpretation corresponding to B. By Theorem 3, S is not τ-satisfied by $\mathscr{I}_B$. But then there exists a ground instance C' of a clause C in S which is not τ-satisfied by $\mathscr{I}_B$. There must exist 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 τ-semantic tree T of the set of clauses S. Then every branch of T contains a failure node, it means that no H-interpretation τ-satisfies S. By Theorem 3, S is τ-unsatisfiable.

4 τ-Resolution

We now devise a resolution inference system, called τ-resolution. The τ-resolution inference system is parameterized with a threshold $\tau > \mathsf{W}$. It can semi-decide whether a formula is τ-unsatisfiable. We prove the soundness and completeness of τ-resolution. The soundness is relatively simple, the completeness proof makes use of τ-semantic tree technique described in Section 3.3.

An *inference rule* has the form:

$$\frac{C_1 \quad C_2 \ldots \quad C_n}{C}$$

where the clauses $C_1, C_2 \ldots, C_n$ are the premises and C is the conclusion. An inference rule is τ-sound iff its conclusion is a τ-consequence of its premises. That is, $\{C_1, C_2 \ldots, C_n\} \models_\tau C$.

Definition 18.*Define the τ-resolution rule as follows:*

$$\frac{\Gamma_1 \vee A_1^{\lambda_1} \quad A_2^{\lambda_2} \vee \Gamma_2}{(\Gamma_1 \vee \Gamma_2)\gamma}$$

where

$$\begin{cases} \lambda_1 \cap \lambda_2 \leq -\tau \\ \lambda_1 \cup \lambda_2 \geq \tau \\ \gamma \textit{ is the most general unifier of } A_1 \textit{ and } A_2 \end{cases}$$

$(\Gamma_1 \vee \Gamma_2)\gamma$ is called a τ-resolvent of $\Gamma_1 \vee A_1^{\lambda_1}$ and $A_2^{\lambda_2} \vee \Gamma_2$.

Theorem 4.*The τ-resolution rule 18 is τ-sound.*

*Proof.*Let $\mathscr{I} = (\mathsf{M}, \sigma)$ be an interpretation. We need to prove that if $\mathscr{I}((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2)) \geq \tau$ then $\mathscr{I}((\Gamma_1 \vee \Gamma_2)\gamma) \geq \tau$.

It is obvious that $\mathscr{I}((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2)) \geq \tau$ implies

$$\mathscr{I}(((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2)\gamma)) \geq \tau$$

We have that

$$\begin{aligned} &\mathscr{I}(((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2))\gamma) \\ &= \mathscr{I}(((\Gamma_1 \wedge A_2^{\lambda_2}) \vee (\Gamma_1 \wedge \Gamma_2) \vee (A_1^{\lambda_1} \wedge A_2^{\lambda_2}) \vee (A_1^{\lambda_1} \wedge \Gamma_2))\gamma) \\ &= \mathscr{I}((\Gamma_1 \wedge A_2^{\lambda_2})\gamma \vee (\Gamma_1 \wedge \Gamma_2)\gamma \vee (A_1^{\lambda_1} \wedge A_2^{\lambda_2})\gamma \vee (A_1^{\lambda_1} \wedge \Gamma_2)\gamma) \\ &= \mathscr{I}((\Gamma_1 \wedge A_2^{\lambda_2})\gamma) \vee \mathscr{I}((\Gamma_1 \wedge \Gamma_2)\gamma) \vee \mathscr{I}((A_1^{\lambda_1} \wedge A_2^{\lambda_2})\gamma) \vee \mathscr{I}((A_1^{\lambda_1} \wedge \Gamma_2)\gamma) \end{aligned}$$

It is easy to see that:

–$\mathscr{I}((\Gamma_1 \wedge A_2^{\lambda_2})\gamma) \leq \mathscr{I}(\Gamma_1\gamma) \leq \mathscr{I}((\Gamma_1 \vee \Gamma_2)\gamma)$,
–$\mathscr{I}((\Gamma_1 \wedge \Gamma_2)\gamma) \leq \mathscr{I}((\Gamma_1 \vee \Gamma_2)\gamma)$,
–$\mathscr{I}((A_1^{\lambda_1} \wedge A_2^{\lambda_2})\gamma) \leq -\tau$, and
–$\mathscr{I}((A_1^{\lambda_1} \wedge \Gamma_2)\gamma) \leq \mathscr{I}(\Gamma_2\gamma) \leq \mathscr{I}((\Gamma_1 \vee \Gamma_2)\gamma)$.

Now by contradiction assume that $\mathscr{I}((\Gamma_1 \vee \Gamma_2)\gamma < \tau$. Then we have $\mathscr{I}((\Gamma_1 \wedge A_2^{\lambda_2})\gamma) \vee \mathscr{I}((\Gamma_1 \wedge \Gamma_2)\gamma) \vee \mathscr{I}((A_1^{\lambda_1} \wedge A_2^{\lambda_2})\gamma) \vee \mathscr{I}((A_1^{\lambda_1} \wedge \Gamma_2)\gamma) < \tau$ or equivalently $\mathscr{I}(((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2))\gamma) < \tau$
which contradicts with the fact that

$$\mathscr{I}(((\Gamma_1 \vee A_1^{\lambda_1}) \wedge (A_2^{\lambda_2} \vee \Gamma_2)\gamma)) \geq \tau$$

This completes the proof of the theorem.

Definition 19.*Define the τ-factoring rule as follows:*

$$\frac{A_1^{\lambda} \vee A_2^{\lambda} \vee \Gamma}{(A_1^{\lambda} \vee \Gamma)\gamma}$$

where γ is the most general unifier of A_1 and A_2.
$(A_1^{\lambda} \vee \Gamma)\gamma$ is called a factor of $A_1^{\lambda} \vee A_2^{\lambda} \vee \Gamma$.

Theorem 5.*The factoring rule 19 is sound.*

*Proof.*Straightforward.

Definition 20.*A τ-resolution derivation is a sequence of the form*

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

where

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

Definition 21.*A τ-empty clause is of the form $A_1^{\lambda_1} \vee \ldots \vee A_n^{\lambda_n}$ where A_i is an atom and $-\tau < \lambda_i < \tau$ for $i = \{1, \ldots, n\}$.*

Proposition 6.*A τ-empty clause is not τ-satisfiable.*

*Proof.*Let $A_1^{\lambda_1} \vee \ldots \vee A_n^{\lambda_n}$ where A_i is an atom and $-\tau < \lambda_i < \tau$ for $i = \{1, \ldots, n\}$ be a τ-empty clause. Let $\mathscr{I} = (\mathsf{M}, \sigma)$ be an interpretation. By definition of formula interpretation we have

$$\begin{aligned}[|A_1^{\lambda_1} \vee \ldots \vee A_n^{\lambda_n}|]_{\sigma}^{\mathsf{M}} &= [|A_1^{\lambda_1}|]_{\sigma}^{\mathsf{M}} \cup \ldots \cup [|A_n^{\lambda_n}|]_{\sigma}^{\mathsf{M}} \\ &= \lambda_1 \otimes [|A_1|]_{\sigma}^{\mathsf{M}} \cup \ldots \cup \lambda_n \otimes [|A_n|]_{\sigma}^{\mathsf{M}}\end{aligned}$$

By definition of $\otimes$ and the fact that $-\tau < \lambda_i < \tau$ for $i = \{1, \ldots, n\}$, we have $\lambda_i \otimes [|A_i|]_{\sigma}^{\mathsf{M}} < \tau$ for $i = \{1, \ldots, n\}$. This implies $\lambda_1 \otimes [|A_1|]_{\sigma}^{\mathsf{M}} \cup \ldots \cup \lambda_n \otimes [|A_n|]_{\sigma}^{\mathsf{M}} < \tau$,
which also means $[|A_1^{\lambda_1} \vee \ldots \vee A_n^{\lambda_n}|]_{\sigma}^{\mathsf{M}} < \tau$
In other words $A_1^{\lambda_1} \vee \ldots \vee A_n^{\lambda_n}$ is not τ-satisfiable.

We use $\square^{\tau}$ to denote a τ-empty clause. The empty clause is the clause that does not contains any literals. The empty clause is a τ-empty clause. We use $\square$ to denote the empty clause.

Theorem 6(Soundness). *Let $S_0, \ldots, S_i, \ldots$ be a τ-resolution derivation. If S_n contains a τ-empty clause (for some $n = 0, 1, ...$), then S_0 is τ-unsatisfiable.*

*Proof.*By Theorems 4 and 5, S_i and S_{i+1} are τ-equisatisfiable. This implies that if S_n contains a τ-empty clause, which also means S_n is τ-unsatisfiable, then S_0 is τ-unsatisfiable.

Lemma 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.*Let $C_1' = \Gamma_1' \vee (A_1')^{\lambda_1'}, C_2' = \Gamma_2' \vee (A_2')^{\lambda_2'}$, and γ be a m.g.u of A_1', A_2'. Let σ be a substitution such that $C_1' = C_1\sigma, C_2' = C_2\sigma$ and $C_1 = \Gamma_1 \vee A_1^{\lambda_1}, C_2 = \Gamma_2 \vee A_2^{\lambda_2}$. By resolution rule 18, $C' = (\Gamma_1' \vee \Gamma_2')\gamma = (\Gamma_1 \vee \Gamma_2)\gamma o\sigma$ 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 $C' = (\Gamma_1 \vee \Gamma_2)\gamma o\sigma$ is an instance of $C = (\Gamma_1 \vee \Gamma_2)\theta$ (or of a factor of $C = (\Gamma_1 \vee \Gamma_2)\theta$).

Theorem 7(Completeness). *Let $S_0, \ldots, S_i, \ldots$ be a τ-resolution derivation. If S_0 is τ-unsatisfiable then there exists S_k containing a τ-empty clause.*

*Proof.*According to Lemma 2 if S_0 is τ-unsatisfiable, then for every τ-semantic tree T_0 of S_0 there is a corresponding finite closed tree T_0'. By Lemma 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'^{α_1} and L'^{α_2} where $\alpha_1 \geq \tau$ and $\alpha_2 \leq -\tau$.

Resolving C_1' and C_2', we obtain the clause C' not containing L', and C' is failed at the node N. By Lemma 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 τ-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 a τ-empty clause $\square^{\tau}$ must be in S_k'. By Lemma 3, S_k contains $\square^{\tau}$.

*Example 6.*Let $AX = (LH_p(G), G, H, \leq, \cup, \cap, -, \Rightarrow)$ be a linguistic truth domain where

–$G = \{\bot, \mathsf{F}, \mathsf{W}, \mathsf{T}, \top\}$, where $\mathsf{T} = \mathsf{True}$, $\mathsf{F} = \mathsf{False}$;
–$\bot < \mathsf{F} < \mathsf{W} < \mathsf{T} < \top$;
–F, T are the negative and positive primary generator, respectively;

– $\bot, \mathsf{W}, \top$ are the smallest, neutral, biggest elements, respectively;
– $H^+ = \{\mathsf{V},\mathsf{R}\}$, $H^- = \{\mathsf{P},\mathsf{ML}\}$, with the ordering $\mathsf{R} < \mathsf{V}$ and $\mathsf{ML} < \mathsf{P}$, where $\mathsf{V} = \mathsf{Very}$, $\mathsf{R} = \mathsf{Rather}$, $\mathsf{P} = \mathsf{Possibly}$, $\mathsf{ML} = \mathsf{MoreOrLess}$.

Consider the following set of clauses:

1. $A(x)^{\mathsf{RF}} \vee B(z)^{\mathsf{RF}} \vee C(x)^{\mathsf{PT}}$
2. $C(y)^{\mathsf{RF}} \vee D(y)^{\mathsf{VRT}}$
3. $C(t)^{\mathsf{VVT}} \vee E(t, f(t))^{\mathsf{RF}}$
4. $D(a)^{\mathsf{RF}}$
5. $E(a,u)^{\mathsf{T}}$
6. $F(a)^{\mathsf{PT}}$

where a is a constant symbol and x, y, z, t, u are variables.

Let us consider the case $\tau = \mathsf{RT}$. Then the clause $E(a,u)^{\mathsf{T}}$ is a RT-empty clause. We conclude that the given set of clauses is not RT-satisfiable. Now consider the case $\tau = \mathsf{T}$. Then the clause $F(a)^{\mathsf{PT}}$ is a T-empty clause. We conclude that the given set of clauses is not T-satisfiable.

Now consider the case $\tau = \mathsf{PT}$, then we have the following τ-resolution inferences

$$\frac{C(y)^{\mathsf{RF}} \vee D(y)^{\mathsf{VRT}} \qquad C(t)^{\mathsf{VVT}} \vee E(t, f(t))^{\mathsf{RF}}}{D(t)^{\mathsf{VRT}} \vee E(t, f(t))^{\mathsf{RF}}} [t/y]$$

$$\frac{D(t)^{\mathsf{VRT}} \vee E(t, f(t))^{\mathsf{RF}} \qquad D(a)^{\mathsf{RF}}}{E(a, f(a))^{\mathsf{RF}}} [a/t]$$

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

A PT-empty clause is derived, we conclude that the initial clause set is PT-unsatisfiable. Actually the initial clause set is even τ-unsatisfiable for all $\tau > \mathsf{W}$, since the resolution procedure drives the empty clause from the latter.

5 Approximate Reasoning

In two-valued logics, we know that a formula is valid if and only if its negation is unsatisfiable. This property is used when one wants to check entailments or prove theorems. That is if we want to prove that ϕ entails φ then we use resolution to derive the empty clause from $\phi \wedge \neg\varphi$. However, this schema does not directly work in our setting here. Indeed, we only have that a formula is τ-valid if and only if the truth value of its negation is less than or equal to $-\tau$ under every interpretation. But τ-resolution assumes that $\tau > \mathsf{W}$, which implies $-\tau \leq \mathsf{W}$, and $(-\tau)$-resolution can not be used.

Fortunately, if $\tau = \mathsf{W}$ then the entailment checking by refutation applies here in our framework. However lowering the threshold τ to W also means that the level of vagueness of formulas will likely be lowered. For example the resolution inference

$$\frac{A^{\mathsf{SlightlyTrue}} \qquad A^{\mathsf{SlightlyFalse}}}{\Box}$$

involves two clauses with low level of vagueness, i.e., the truth value is at most SlightlyTrue, while the resolution inference

$$\frac{A^{\mathsf{VeryTrue}} \qquad A^{\mathsf{VeryFalse}}}{\Box}$$

involves two clauses with higher level of vagueness, i.e., the truth value is at most VeryTrue. We say that the latter inference is more *confident* than the former. In this case we successfully derive the empty clause and we say that the latter proof of the empty clause is more confident than the former. In practice more confident inferences and proofs should be preferred. Below we formalize the notion of confidence in our resolution framework.

Definition 22. *Let δ be an element of $LH_p(G)$ and C be a clause. The clause C with confidence δ is the pair (C, δ). The same clauses with different reliabilities are called variants. That is (C, δ) and (C, δ') are called a variant of each other.*

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

$$\frac{(C_1, \delta_1) \quad (C_2, \delta_2) \ \ldots \ (C_n, \delta_n)}{(C, \delta)}$$

where $(C_1, \delta_1) \quad (C_2, \delta_2) \ \ldots \ (C_n, \delta_n)$ are premises and (C, δ) is the conclusion. We say that δ is the confidence of R, provided that $\delta \leq \delta_i$ for $i = 1, 2, \ldots, n$. The soundness of an inference is defined as for inferences without confidence.

The resolution rule for clauses is adapted as follows:

$$\frac{(\Gamma_1 \vee A_1^{\lambda_1}, \delta_1) \quad (A_2^{\lambda_2} \vee \Gamma_2, \delta_2)}{((\Gamma_1 \vee \Gamma_2)\gamma, \delta)}$$

where

$$\begin{cases} \lambda_1 \cap \lambda_2 < \mathsf{W} \\ \lambda_1 \cup \lambda_2 \geq \mathsf{W} \\ \gamma \text{ is the most general unifier of } A_1 \text{ and } A_2 \\ \delta = \delta_1 \cap \delta_2 \cap (\lambda_1 \cup \lambda_2) \cap (-\lambda_1 \cup -\lambda_2) \end{cases}$$

$((\Gamma_1 \vee \Gamma_2)\gamma, \delta)$ is called a τ-resolvent of $(\Gamma_1 \vee A_1^{\lambda_1}, \delta_1)$ and $(A_2^{\lambda_2} \vee \Gamma_2, \delta_2)$.

The factoring rule for clauses is adapted as follows:

$$\frac{(A_1^{\lambda} \vee A_2^{\lambda} \vee \Gamma, \delta)}{((A_1^{\lambda} \vee \Gamma)\gamma, \delta)}$$

where γ is the most general unifier of A_1 and A_2. $((A_1^{\lambda} \vee \Gamma)\gamma, \delta)$ is called a factor of $(A_1^{\lambda} \vee A_2^{\lambda} \vee \Gamma, \delta)$.

The notion of resolution derivation for clauses with confidence can be adapted as follows. A resolution derivation is a sequence of the form

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

where

– S_i is a set of clauses with confidence (for $i = 1, \ldots, n$), and
– $S_{i+1} = S_i \cup (C, \delta)$, and $(C, \delta) \notin S_i$, and (C, δ) is the conclusion of a resolution inference with premises from S_i or of a factoring with premise from S_i.

*Example 7.*Consider again the linguistic truth domain in Example 6. Consider the set of clauses:

1.$(A(x)^{\mathsf{VF}} \vee B(z)^{\mathsf{VF}} \vee C(x)^{\mathsf{PT}}, \mathsf{T})$
2.$(C(y)^{\mathsf{SF}} \vee D(y)^{\mathsf{VPT}}, \mathsf{T})$
3.$(C(t)^{\mathsf{VVT}} \vee E(t, f(t))^{\mathsf{VF}}, \mathsf{T})$
4.$(D(a)^{\mathsf{VF}}, \mathsf{T})$
5.$(E(a,u)^{\mathsf{T}}, \mathsf{T})$

At the beginning, the confidence of each clause is assigned to T.

We have the following resolution inferences

$$\frac{(C(y)^{\mathsf{SF}} \vee D(y)^{\mathsf{VPT}}, \mathsf{T}) \qquad (C(t)^{\mathsf{VVT}} \vee E(t, f(t))^{\mathsf{VF}}, \mathsf{T})}{(D(t)^{\mathsf{VPT}} \vee E(t, f(t))^{\mathsf{VF}}, \mathsf{ST})}\ [t/y]$$

$$\frac{(D(t)^{\mathsf{VPT}} \vee E(t, f(t))^{\mathsf{VF}}, \mathsf{ST}) \qquad (D(a)^{\mathsf{VF}}, \mathsf{T})}{(E(a, f(a))^{\mathsf{VF}}, \mathsf{ST})}\ [a/t]$$

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

The empty clause is derived, we conclude that the initial clause set is unsatisfiable and the confidence of the proof of unsatisfiability is $\mathsf{SlightlyTrue}$.

A *resolution proof* of a clause C from a set of clauses S consists of repeated application of the resolution rule to derive the clause C from the set S. If C is the empty clause then the proof is called a *resolution refutation*. We will represent resolution proofs as *resolution trees*. Each tree node is labeled with a clause. There must be a single node that has no child node, labeled with the conclusion clause, we call it the root node. All nodes with no parent node are labeled with clauses from the initial set S. All other nodes must have two parents and are labeled with a clause C such that

$$\frac{(C_1, \delta_1) \qquad (C_2, \delta_2)}{(C, \delta)}$$

where $(C_1, \delta_1), (C_2, \delta_2)$ are the labels of the two parent nodes. If there is a resolution proof of a clause (C, δ), then the resolution proof and representing resolution tree are said to have the confidence δ.

Different resolution proofs may give the same the conclusion clause with different confidences. The following example illustrates this.

*Example 8.*Consider again the linguistic truth domain in Example 6. Consider the set of clauses:

1.$(A^{\mathsf{T}}, \mathsf{T})$
2.$(A^{\mathsf{VF}} \vee B^{\mathsf{F}}, \mathsf{T})$
3.$(B^{\mathsf{VT}}, \mathsf{T})$
4.$(B^{\mathsf{VF}} \vee C^{\mathsf{F}}, \mathsf{T})$
5.$(C^{\mathsf{AT}}, \mathsf{T})$

At the beginning we assign each clause to the confidence T. Then the set of clauses

$$\{A^{\mathsf{VF}} \vee B^{\mathsf{F}}, \mathsf{T}), (B^{\mathsf{VT}}, \mathsf{T}), (A^{\mathsf{T}}, \mathsf{T})\}$$

will give the following resolution refutation with the confidence T

$$\frac{\dfrac{(A^{\mathsf{VF}} \vee B^{\mathsf{F}}, \mathsf{T}) \qquad (B^{\mathsf{VT}}, \mathsf{T})}{(A^{\mathsf{VF}}, \mathsf{T})} \qquad (A^{\mathsf{T}}, \mathsf{T})}{(\Box, \mathsf{T})}$$

The other set of clauses

$$\{(B^{\mathsf{VF}} \vee C^{\mathsf{F}}, \mathsf{T}), (B^{\mathsf{VT}}, \mathsf{T}), (C^{\mathsf{AT}}, \mathsf{T})\}$$

will give the following resolution refutation with the confidence AT

$$\frac{\dfrac{(B^{\mathsf{PF}} \vee C^{\mathsf{F}}, \mathsf{T}) \qquad (B^{\mathsf{VT}}, \mathsf{T})}{(C^{\mathsf{F}}, \mathsf{PT})} \qquad (C^{\mathsf{AT}}, \mathsf{T})}{(\Box, \mathsf{PT})}$$

Clearly the former resolution refutation has a greater confidence than the latter resolution refutation.

Example 8 raises a natural question whether we can find the refutation with the maximal confidence. Below we present a resolution strategy, called Δ-*strategy*, which guarantees that the resolution proof of each clause has the maximal confidence.

A set of clauses S is said to be Δ- *saturated* iff for every resolution inference with premises in S the conclusion of this inference is a variant with smaller or equal reliability of some clause in S. That is for every resolution inference

$$\frac{(C_1, \delta_1) \qquad (C_2, \delta_2)}{(C, \delta)}$$

where $(C_1, \delta_1), (C_2, \delta_2) \in S$, there is some clause $(C, \delta') \in S$ such that $\delta \leq \delta'$.

A Δ-strategy derivation is a sequence of the form

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

where each S_i is a set of clauses, and

– S_{i+1} is obtained by adding the conclusion of a resolution inference with premises with maximal confidences from S_i, that is $S_{i+1} = S_i \cup \{(C, \delta)\}$, where (C, δ) is the conclusion of the resolution inference

$$\frac{(C_1, \delta_1) \qquad (C_2, \delta_2)}{(C, \delta)}$$

$(C_1, \delta_1), (C_2, \delta_2) \in S_i$ and there are not any clauses $(C_1, \delta_1'), (C_2, \delta_2') \in S_i$ such that $\delta_1' > \delta_1$ and $\delta_2' > \delta_2$, or

– S_{i+1} is obtained by removing a variant with smaller confidence, that is $S_{i+1} = S_i \setminus \{(C,\delta)\}$ where $(C,\delta) \in S_i$ and there is some $(C,\delta') \in S_i$ such that $\delta < \delta'$.

Define *the limit of a derivation* $S_0,\ldots,S_i,\ldots$

$$S_\infty = \bigcup_{i\geq 0}\bigcap_{j\geq i} S_j$$

Similar to the standard τ-resolution procedure, the Δ-strategy procedure is sound and complete. The proofs are exactly the same.

Theorem 8.*Let $S_0,\ldots,S_i,\ldots$ be a Δ-strategy derivation. Then S_0 is W-unsatisfiable if and only if S_n contains the empty clause (for some $n = 0,1,\ldots$).*

The most interesting property of the Δ-strategy procedure is that we can show that it gives resolution proofs with maximal confidence.

Lemma 4.*Consider the following resolution inferences:*

$$\frac{(\Gamma_1 \vee A_1^{\lambda_1},\delta_1) \quad (A_2^{\lambda_2} \vee \Gamma_2,\delta_2)}{((\Gamma_1 \vee \Gamma_2)\gamma,\delta)}$$

$$\frac{(\Gamma_1 \vee A_1^{\lambda_1},\delta_1) \quad (A_2^{\lambda_2} \vee \Gamma_2,\delta_2')}{((\Gamma_1 \vee \Gamma_2)\gamma,\delta')}$$

Then, $\delta_2' > \delta_2$ implies $\delta' \geq \delta$.

*Proof.*It is easy to see that if $\delta_2' > \delta_2$ then

$$\delta' = \delta_1 \cap \delta_2' \cap (-\lambda_1 \cup -\lambda_2) \cap (\lambda_1 \cup \lambda_2)$$

$$\geq \delta_1 \cap \delta_2 \cap (-\lambda_1 \cup -\lambda_2) \cap (\lambda_1 \cup \lambda_2) = \delta$$

Lemma 5.*Let $S_0,\ldots,S_i,\ldots$ be a Δ-strategy derivation, and S_∞ be the limit of the derivation. Then S_∞ is Δ-saturated.*

*Proof.*By contradiction assume that S_∞ is not Δ-saturated. Then there must be a resolution inference

$$\frac{(C_1,\delta_1) \quad (C_2,\delta_2)}{(C,\delta)}$$

where $(C_1,\delta_1),(C_2,\delta_2) \in S_\infty$, there is not any clause $(C,\delta') \in S_\infty$ such that $\delta \leq \delta'$. By definition of Δ-strategy derivation, either (C,δ) is in S_∞ or there must be a clause (C,δ'') in S_i for some $i = 0,1,\ldots$ such that $\delta \leq \delta''$, this also means that (C,δ) is removed from S_j for some $j \geq i$. In both cases, we have a contradiction.

Theorem 9.*Let $S_0,\ldots,S_i,\ldots$ be a Δ-strategy derivation, and S_∞ be the limit of the derivation. Then for each clause (C,δ) in S_∞, there is not any other resolution proof of the clause (C,δ') from S_0 such that $\delta' > \delta$.*

*Proof.*By contradiction, suppose that for some clause (C,δ) in S_∞, there exists a resolution proof of (C,δ') from S_0 such that $\delta' > \delta$. Let (C_1,δ_1) and (C_2,δ_2) be the two parents of (C,δ') in such a resolution proof of (C,δ'). We have that (C_1,δ_1) and (C_2,δ_2) cannot be both in S_∞ because otherwise an inference with these two clauses as premisses would give (C,δ') in S_∞. Without lost of generality, we can assume that (C_1,δ_1) is not in S_∞. That also means there is a clause (C_1,δ_1') in S_∞ such that $\delta_1' > \delta_1$. By Lemma 5, S_∞ is Δ-saturated. According to Lemma 4, the inference with premisses (C_1,δ_1') and (C_2,δ_2) gives us the conclusion (C,δ''), with $\delta'' > \delta'$. This contradicts with the fact that S_∞ is Δ-saturated. This completes the proof of the theorem.

*Example 9.*Consider again Example 8. Applying the Δ-strategy we get the following saturated set of clauses

1.$(A^{\mathsf{T}},\mathsf{T})$
2.$(A^{\mathsf{VF}} \vee B^{\mathsf{F}},\mathsf{T})$
3.$(B^{\mathsf{VT}},\mathsf{T})$
4.$(B^{\mathsf{PF}} \vee C^{\mathsf{F}},\mathsf{T})$
5.$(C^{\mathsf{AT}},\mathsf{T})$
6.$(B^{\mathsf{F}},\mathsf{T})$
7.$(A^{\mathsf{VF}},\mathsf{VT})$
8.$(C^{\mathsf{F}},\mathsf{VT})$
9.$(B^{\mathsf{PF}},\mathsf{AT})$
10.$(\Box,\mathsf{T})$

The initial set of clauses is unsatisfiable, and the resolution refutation is

$$\dfrac{\dfrac{(A^{\mathsf{T}},\mathsf{T}) \quad (A^{\mathsf{VF}} \vee B^{\mathsf{F}},\mathsf{T})}{(B^{\mathsf{F}},\mathsf{T})} \quad (B^{\mathsf{VT}},\mathsf{T})}{(\Box,\mathsf{T})}$$

6 Related Work

The starting point of our investigation is[21,15]. The main difference between LPL and fuzzy operator logic of [15] lies in the truth domain. We use a linguistic truth domain instead of the real interval $[0,1]$ of fuzzy logic as in [15]. This allows us to manipulate directly linguistic terms in describing vague statements and mechanizing the human reasoning under vagueness. We differ from [21] in both algebraic and deductive aspects. Our linguistic truth domain is a refined hedge algebra while [21] uses a linear hedge algebra. It is worth underlying that in practice we often have to deal with incomparable hedges, such as MoreOrLess and Approximately, and incomparable linguistic truth values, such as MoreOrLessTrue and ApproximatelyTrue. In these situations our logic can be used but the one of [21] cannot. On the other hand, our inference system is based on resolution, while [21] uses hedge moving rules on arbitrary linguistic terms. Our resolution inference system is sound and complete but the inference system in [21] is only consistent.

The resolution for fuzzy logics has been an active research direction [18,31,23,22,15]. In [31] Lee presented a resolution procedure for fuzzy logics and showed that the equisatisfiability of logical clauses in fuzzy logics and two-valued logics. Shen et al. [23] went a step further by giving a fuzzy refutation complete resolution procedure in which the premises and conclusions, as well as the inferential results are all fuzzy. Weigert et al. [15] presented an approach to fuzzy logic and reasoning under vagueness using the resolution principle based on the fuzzy operator. Smutná and Vojtás [22] gave a sound resolution for multiple-valued logics with arbitrary connectives and graded information, and a resolution truth function to evaluate the truth value of the resolvent of a resolution inference. Habiballa [18] presented a sound and complete resolution inference system for fuzzy predicate logic with evaluated syntax. Our resolution framework is heavily based on the one of [15], but extends it with the concept of proof confidence to capture the approximate nature of resolution inferences.

Notable works in using hedge algebra for linguistic reasoning are [32,21]. Nguyen et al. [21] presented a logics whose truth domain is based on linear symmetric hedge algebras along with an inference system consisting of hedge moving rules. Le et al. [32] proposed a sound and complete fuzzy linguistic logic programming whose truth domain is based on linear finite symmetric hedge algebras. The fuzzy linguistic logic programming only works with a restricted class of rules, not as expressive as our class of formulae. Since CNF transformation is difficult in predicate fuzzy logic with evaluated syntax, the main advantage of non-clausal resolution is that it does not require CNF transformation. However, the evaluated syntax is rather restricted, in the sense that truth values are over formulae and cannot be distributed over subformulae. Beside it seems difficult to have an ordered version of non-clausal resolution for faster proof search as in [6]. Notice that all of the works [32,21] make use of linear symmetric hedge algebras as the linguistic truth domains. Here we consider the class of refined hedge algebras which includes the class of linear hedge algebras. In this respect, LPL is more expressive than the ones in [32,21].

In [37,36,35] we considered a more restricted resolution inference system for a more restricted class of formulae where truth operators are only over atoms. Here we generalize the results in [36,37,35] in two ways. First, we consider a syntactically more complex class of formulae where truth operators are over formulae, and can be distributed over logical connectives and truth operators during CNF transformation. As an example, the formula

$$Smart(John)^{\mathsf{False}} \vee WorkHard(John)^{\mathsf{RatherFalse}}$$

is allowed in [37,36], but the following formula is not

$$(Smart(John)^{\mathsf{VeryTrue}} \wedge WorkHard(John)^{\mathsf{RatherTrue}})^{\mathsf{False}}$$

Second, the τ-resolution procedure is parameterized with a truth degree threshold $\tau \geq \mathsf{W}$, in contrast with [37,36] which only consider the case $\tau = \mathsf{W}$. Consequently our LPL and resolution inference system are more expressive and flexible than the ones in [37,36].

It is worth mentioning that algebraic and logical aspects of fuzzy logics have been for longtime an active research direction (see e.g., [17,4,16] for more detailed surveys). This line of research is concerned with fuzzy logics in the narrow sense where one is interested in proof theories for fuzzy logics. Here we consider fuzzy logics in the broad sense where we are interested in the human reasoning under vague statements in natural languages. Especially the degree of vagueness is expressed by means of linguistic truth terms, and the reasoning under vagueness is based on resolution. Also algebraic [24,26,25] and logical [8,10,27,19,9] approaches to linguistic truth have been attractive research topics. Here we adopt the algebraic approach to linguistic truth. This allows us to treat the problem of the human reasoning under vagueness as the problem of approximate reasoning with LPL. And the latter can be handled smoothly because many interesting results on approximate reasoning with multiple-valued and fuzzy logics are transferable to LPL, as we have shown.

7 Conclusion

We have presented the LPL for the problem of *describing* vague statements in natural languages, along with the τ-resolution inference system for the problem of *mechanizing* the human reasoning under vagueness. LPL has the following features: the truth domain is a refined hedge algebra generated by a set of truth generators and a set of hedges; the syntax is based on the one of two-valued predicate logics but augmented with truth operators; the semantics generalizes the one of two-valued logics; the syntax and semantics properties make sure that every formula has an equisatisfiable CNF formula, and a CNF transforming algorithm has been given. The τ-resolution inference system, parameterized with a threshold τ, is sound and complete. We have shown how τ-resolution is used to check τ-satisfiability of sets of clauses. We have studied the special case where $\tau = \mathsf{W}$, namely the neutral element, and shown how τ-resolution is used to check entailments or to prove theorems. The τ-resolution inferences have a level of confidence, and lowering the threshold τ to W makes them less confident. We have proposed the Δ-strategy to guarantee that inferences performed are always the most confident. The Δ-strategy is, as the standard τ-resolution, sound and complete.

There are two main lines of future work. First we intend to study an ordered version of τ-resolution for faster proof search along the line of [6]. Ordering and redundancy elimination techniques based on ordering have shown to be crucial for the performance of

resolution theorem proving. It would be interesting to implement and experiment such an ordered τ-resolution. Second we plan to apply our approach here to description logics which are commonly used in knowledge representation and ontologies, along the line of [1,2,3]. The idea is to use symmetric refined hedge algebras as the truth domains of the description logics, and tableaux as the automated reasoning method. This line of research seems to be promising because vagueness is unavoidable when modeling many real-world applications, especially in medical informatics.

Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.04-2013.21.

References

[1] S. Borgwardt and Rafael Peñaloza, *Description Logics over Lattices with Multi-Valued Ontologies*, IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, (2011), 768–773.

[2] Y. Jiang, Y. Tang, J. Wang, P. Deng and S. Tang, *Expressive fuzzy description logics over lattices*, Knowl.-Based Syst, v. 23, No. 2,(2010), 150–161.

[3] U. Straccia, *Description Logics over Lattices*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, v. 14, No. 1,(2006), 1–16.

[4] H. Rasiowa, *An algebraic approach to non-classical logics*, North Holland Publishing, (1974).

[5] G. Birkhoff, *Lattice Theory*, Providence, RhodeIsland, v. 14, No. 1, (1974).

[6] Bachmair, L. and Ganzinger, H., *Resolution Theorem Proving*, North Holland, v. 1, (2001), 19–100.

[7] G. Lakoff, *Hedges: a study in meaning criteria and the logic of fuzzy concepts*, Journal of Philosophical Logic, v. 2, (1973), 458-508.

[8] F. Esteva, L. Godo and C. Noguera, *A logical approach to fuzzy truth hedges*, Inf. Sci., v. 232, (2013), 366–385.

[9] Vilém Vychodil, *Truth-depressing hedges and BL-logic*, Fuzzy Sets and Systems, v. 157, No. 15, (2006), 2074–2090.

[10] Petr Hájek, *On very true*, Fuzzy Sets and Systems, v. 124, No. 3, (2001), 329–333.

[11] J. J. Lu, Neil V. Murray and E. Rosenthal, *A Framework for Automated Reasoning in Multiple-Valued Logics*, Journal of Automated Reasoning, v. 21, No. 1, (1998), 39–67.

[12] Baaz, Matthias and Fermüller, Christian G. and Salzer, Gernot, *Automated Deduction for Many-valued Logics*, Journal of Symbolic Computation, isbn 0-444-50812-0, (2001), 1355–1402.

[13] Reiner Hähnle, *Short Conjunctive Normal Forms in Finitely Valued Logics*, Journal of Logic Computation, v. 4, No. 6, (1994), 905–927.

[14] Reiner Hähnle, *Exploiting Data Dependencies in Many-Valued Logics*, Journal of Applied Non-Classical Logics, v. 6, No. 1, (1996), 49–69.

[15] T.J. Weigert, J. J. P. Tsai and X. Liu, *Fuzzy Operator Logic and Fuzzy Resolution*, Journal of Automated Reasoning, v. 10, No. 1, (1993), 59–78.

[16] G. M. and N. Olivetti and D. Gabbay, *Proof theories for Fuzzy Logics*, Kluwer, (2009).

[17] Hájek, P., *Metamathematics of Fuzzy Logic*, Springer, (1998).

[18] H. Habiballa, *Non-clausal resolution and fuzzy logic*, Non-clausal resolution and fuzzy logic, v. 4, No. 3, (2011).

[19] Novák, V., *A Comprehensive Theory of Trichotomous Evaluative Linguistic Expressions*, Fuzzy Sets and Systems, v. 159, No. 22, (2008) 2939-2969.

[20] C. Nguyen and V. Huynh, *An algebraic approach to linguistic hedges in Zadeh's fuzzy logic*, Fuzzy Sets and Systems, v. 129, No. 2, (2002) 229-254.

[21] C. Nguyen, D. K. Tran, V. Huynh and H. Nguyen, *Algebras, Linguistic-Valued Logic and Their Application to Fuzzy Reasoning*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, v. 7, No. 4, (1999) 347-361.

[22] Dana Smutná and Peter Vojtás, *Graded many-valued resolution with aggregation*, Fuzzy Sets and Systems, v. 143, No. 1, (2004) 157-168.

[23] Z. Shen, L. Ding and M. Mukaidono, *Fuzzy resolution principle*, Proceedings of 18th International Symposium on Multiple-valued Logics, (1989) 210-215.

[24] N. C. Ho, W. Wechler, *Extended hedge algebras and their application to fuzzy logic*, Fuzzy Sets and Systems, v. 52, No. 3, (1992).

[25] N. C. Ho, W. Wechler, *Hedge Algebras: An Algebraic Approach in Struture of Sets of Linguistic Truth Values*, Fuzzy Sets and Syst. 35, (1990) 281–293.

[26] V. Huynh, Y. Nakamori and T. Murai, *An Algebraic Foundation for Linguistic Reasoning*, Fundam. Inform., v. 78, No. 2, (2007) 271-294.

[27] J. Lai and Y. Xu, *Linguistic truth-valued lattice-valued propositional logic system lP(X) based on linguistic truth-valued lattice implication algebra*, Information Science., v. 180, No. 10, (2010) 1990-2002.

[28] L. A. Zadeh, *The concept of a linguistic variable and its application to approximate reasoning - I*, Information Science, v. 8, No. 3, (1975) 199-249.

[29] L. A. Zadeh, *The concept of a linguistic variable and its application to approximate reasoning - II*, Information Science, v. 8, No. 4, (1975) 301-357.

[30] L. A. Zadeh, *The concept of a linguistic variable and its application to approximate reasoning - III*, Information Science, v. 9, No. 1, (1975) 43-80.

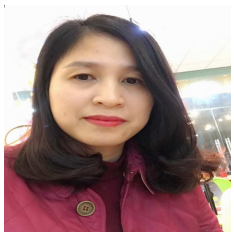
[31] R. C. T. Lee, *Fuzzy Logic and the Resolution Principle*, International Joint Conference on Artificial Intelligence, (1971) 560-567.

[32] V. H. Le, F. L. and D. K. Tran, *Fuzzy linguistic logic programming and its applications*, Theory and Practice of Logic Proramming, v. 9, No. 3, (2009) 309-341.

[33] J. A. Robinson, *A Machine-Oriented Logic Based on the Resolution Principle*, Journal of ACM, v. 12, No. 1, (1965) 23-41.

[34] L. A. Zadeh, *Information and Control*, Journal of ACM, v. 8, No. 3, (1965) 338-353.

[35] M. T. Nguyen and D. K. Tran, *Resolution Method in Linguistic Propositional Logic*, International Journal of Advanced Computer Science and Applications, v. 7, No. 1, (2016) 672-678.
[36] D. Tran, V. Vu, T. Doan and M. Nguyen, *Fuzzy linguistic propositional logic based on refined hedge algebra*, Proceedings of 2013 IEEE International Conference on Fuzzy Systems, ISSN 1098-7584, (2013) 1-8.
[37] D. Tran and M. Nguyen, *Fuzzy linguistic first order logic based on refined hedge algebra*, Proceedings of 2014 International Conference on Fuzzy Systems, Beijing, China, July 6-11, 2014, ISSN 1098-7584, (2014) 1156–1163.



Nguyen Thi-Minh-Tam is currently a Lecturer in the Faculty of Information Technology at Vinh University in Vietnam. Her research interests are logics, automated reasoning, and artificial intelligence. She has contributed research articles in scientific conferences and journals.

Duc Khanh Tran received his Ph.D. degree from Henri Poincare University in France, and did his Postdoc at Max Planck Institute for Informatics in Germany. His research interest includes Logics, Automated Reasoning, and Artificial Intelligence. He has published research articles in reputed scientific conferences and journals. He is currently working at Ho Chi Minh University of Technology in Vietnam, as Head of Research Department.