Reasoning and Experimenting within Zadeh's Fuzzy Propositional Logic

Umberto Straccia
I.E.I. - C.N.R., Pisa (Italy)
straccia@iei.pi.cnr.it

July 6, 2000

Abstract

We present several decision algorithms within the fuzzy propositional logic based on Zadeh's implication operator $p \rightarrow q = \max\{1-p, q\}$, deciding both the fuzzy SAT problem as well as the *best truth value bound* problem, i.e. compute the best truth value bounds of a proposition with respect to a theory. Further, we evaluate all the algorithms by adapting and extending the well know methods for evaluating SAT decision algorithms. We show that both problems present the typical *easy-hard-easy* pattern.

ACM Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic - *Model theory*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving - *Deduction*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods - *Representations*

1 Introduction

Since the introduction of fuzzy sets by Zadeh [17], an impressive work has been carried out around them, not least the numerous studies on fuzzy logics. In fuzzy logics, the notion of *grade of membership* of an element x in an universe U with respect to a fuzzy subset A over U is regarded as the *truth value* of the statement *"x is A"*. In this paper we (re) consider fuzzy logic with truth values in the interval $[0, 1]$ and limit our presentation to the propositional case. This work is originate by [1] where a correct and complete inference system has been presented for fuzzy propositional logic using Zadeh's implication operator. Initially, Pavelka [12] presented its formalisation of fuzzy logic based on Lukasiewicz's implication operator [10, 7], where the logical system allows one to infer the lower-bound for the truth values of a formula. In [6], a formalisation of fuzzy logic using Zadeh's implication was presented, which allows one to infer the upper-bound of a formula. In [6] several inference rules have been identified which were not complete in the sense that they may not always to infer the bets possible upper-bound. Finally, in [1] the ideas of [12] and [6] have been combined to define the notions of both the lower and upper bounds of a formula. In [6] a correct and complete set of inference rules has been presented based on Zadeh's implication which allows us to decide the entailment problem.

In this paper we will continue the work presented in [1] along three directions: (*i*) we present an alternative, correct and complete decision procedure for the satisfiability problem and, thus, for the entailment problem. The decision algorithm is a variant of the classical DPLL (Davis-Putnam-Longemann-Loveland) procedure [3, 4] for classical propositional logic not in conjunctive normal form; (*ii*) we present two new decision algorithms for the *Best Truth Value Bound* (BTVB) problem, i.e. the problem of determining the greatest lower bound and the least upper bound of the truth value of a proposition with respect to a theory (not addressed in [1]); (*iii*) we implemented and evaluated all our algorithms by adapting and extending the well know methods for evaluating SAT decision algorithms (see, e.g. [11]) and show that both the fuzzy SAT problem as well as the BTVB bound problem present a typical *easy-hard-easy* pattern [15].

We proceed as follows. In the next section we introduce syntax, semantics of the fuzzy propositional logic considered and main definitions. In Section 3 we recall some basic properties of the logic, while in Section 4 we present our decision algorithms for the SAT problem and the BTVB problem. In Section 5 we evaluate the algorithms, while Section 6 concludes and presents future research directions.

2 Syntax and semantics

Our logical language has two parts. At the *objective level*, let $\mathcal{L}$ be the language of propositional logic, with connectives $\wedge, \vee$, $\neg$, $\rightarrow$ and the logical constants $\bot$ (false) and $\top$ (true). We will use metavariables $A, B, C, \ldots$ and $p, q, r, \ldots$ for propositions and propositional letters, respectively[1]. $\bot, \top$, letters and their negations are called *literal* (denoted l). As we will see below, propositions will have a truth value in $[0, 1]$.

At the *meta level*, let $\mathcal{L}^M$ be the language of *meta propositions* (denoted by ψ). $\mathcal{L}^M$ consists of *meta atoms*, i.e. expressions of type $(A{\geq}n)$ and $(A{\leq}n)$, where A is a proposition in $\mathcal{L}$ and $n \in [0, 1]$, the connectives $\wedge, \vee$, $\neg$, $\rightarrow$ and the logical constants $\bot$ and $\top$. Essentially, a meta-atom $(A{\leq}n)$ constrains the truth value of A to be less or equal to n (similarly for $\geq$). But, unlike [12] where the truth value of $(A{\leq}n)$ can be any number in $[0, 1]$, in our case $(A{\leq}n)$ will have the truth value 0 or 1.

A *meta letter* is a meta atom the form $(p{\geq}n)$ and $(p{\leq}n)$, where p is a propositional letter. $\bot, \top$, meta letters and their negations are called *meta literal*. A meta proposition is then any $\wedge, \vee$, $\neg$, $\rightarrow$ combination of meta-literal. For instance, $(\neg(r \wedge s{\leq}0.6) \vee (p \vee q{\geq}0.2)) \rightarrow (r \wedge s{\leq}0.6)$ is a meta proposition, while $(p \rightarrow 0.2)$ and $((p{\leq}0.3) \geq 0.4)$ are not. We will use $(A{<}n)$ as a short form of $\neg(A{\geq}n)$ and similarly for $(A{>}n)$; likewise, $(A{=}n)$ is a short form for $(A{\leq}n) \wedge (A{\geq}n)$. The meta letter $(p{\geq}n)$ is *non-trivial* if $n > 0$, and similarly for $(p{\leq}n)$. The meta letter $(p{\geq}1)$ corresponds to the classical letter p (p is true), and $(p{\leq}0)$ corresponds to the classical literal $\neg p$ (p is false). Therefore, $\mathcal{L}^M$ contains $\mathcal{L}$.

The classical definitions of *Negation Normal Form* (NNF), *Conjunctive Normal Form* (CNF) and *Disjunctive Normal Form* (DNF) are easily extended to our context. For instance, a meta proposition ψ in *negation normal form* is an $\wedge, \vee$ combination of meta literal (note that no $\rightarrow$ occurs in ψ); a meta proposition ψ in *conjunctive normal form* is a conjunction of disjunction of meta literal. Similarly for the DNF case.

From a semantics point of view, an interpretation $\mathcal{I}$ is a mapping $(\cdot)^{\mathcal{I}}$ from propositional letters into $[0, 1]$. We extend $\mathcal{I}$ to propositions via

$$
\begin{aligned}
\top^{\mathcal{I}} &= 1 && (1)\\
\bot^{\mathcal{I}} &= 0 && (2)\\
(\neg A)^{\mathcal{I}} &= 1 - A^{\mathcal{I}} && (3)\\
(A \wedge B)^{\mathcal{I}} &= \min\{A^{\mathcal{I}}, B^{\mathcal{I}}\} && (4)\\
(A \vee B)^{\mathcal{I}} &= \max\{A^{\mathcal{I}}, B^{\mathcal{I}}\} && (5)\\
(A \rightarrow B)^{\mathcal{I}} &= \max\{1 - A^{\mathcal{I}}, B^{\mathcal{I}}\} \text{ (Zadeh's implication operator)} && (6)
\end{aligned}
$$

Given an interpretation $\mathcal{I}$ we will assign a boolean truth value in $\{0, 1\}$ to each meta atom in the obvious way: namely,

$$(A{\geq}n)^{\mathcal{I}} = 1, \text{ iff } A^{\mathcal{I}} \geq n \qquad (7)$$

[1]In the following, all metavariables could have an optional subscript and superscript.

$$(A \leq n)^{\mathcal{I}} \quad = \quad 1, \text{ iff } A^{\mathcal{I}} \leq n \tag{8}$$

It is worth noting that neither the relations "$\geq$", and "$\leq$" nor the number n is interpreted as fuzzy. Finally, we assign a boolean truth value to each meta proposition like $(A \geq n_1) \vee (B \leq n_2)$ using the classical method of combining truth values. We say that an interpretation $\mathcal{I}$ *satisfies* a meta proposition ψ if $\mathcal{I}(\psi) = 1$; in that case, we will say that $\mathcal{I}$ is a *model* of ψ.

A *meta theory* (denoted by Σ) is a finite set of meta propositions. Given an interpretation $\mathcal{I}$ and a meta theory Σ, we say that $\mathcal{I}$ *satisfies* Σ if $\mathcal{I}$ satisfies each $\psi \in \Sigma$; in that case we say that $\mathcal{I}$ is a *model* of Σ. We say that a meta theory Σ *entails* a meta proposition ψ if every model of Σ is a model of ψ; this is denoted by $\Sigma \approx\!\!| \psi$. A meta proposition ψ is *valid* if it is entailed by the empty meta theory, i.e. $\emptyset \approx\!\!| \psi$. An example of valid meta proposition is $(A \wedge \neg A \leq 0.5)$. Two propositions A and B are said to be *equivalent* (denoted by $A \equiv B$) if $A^{\mathcal{I}} = B^{\mathcal{I}}$, for each interpretation $\mathcal{I}$. For example, $\neg(A \wedge \neg B)$ is equivalent to $(\neg A) \vee B$. The equivalence of two meta propositions is defined in the same way.

Given a meta theory Σ and a proposition A, it is of interest to compute A's best lower and upper truth value bounds. To this end we define the *least upper bound* and the *greatest lower bound* of A with respect to Σ (written $lub(\Sigma, A)$ and $glb(\Sigma, A)$, respectively) as

$$\begin{aligned} lub(\Sigma, A) &= \inf\{n : \Sigma \approx\!\!| (A \leq n) \text{ or } \Sigma \approx\!\!| (A < n)\} && (9) \\ glb(\Sigma, A) &= \sup\{n : \Sigma \approx\!\!| (A \geq n) \text{ or } \Sigma \approx\!\!| (A > n)\} && (10) \end{aligned}$$

Notice that, e.g. $\Sigma \approx\!\!| (A \geq n)$ iff $glb(\Sigma, A) \geq n$ and $\Sigma \approx\!\!| (A \leq n)$ iff $lub(\Sigma, A) \geq n$. Furthermore, since $\Sigma \approx\!\!| (A < n)$ implies $\Sigma \approx\!\!| (A \leq n)$ and $\Sigma \approx\!\!| (A > n)$ implies $\Sigma \approx\!\!| (A \geq n)$, it follows that the definitions of $lub(\Sigma, A)$ and $glb(\Sigma, A)$ can be simplified to $lub(\Sigma, A) = \inf\{n : \Sigma \approx\!\!| (A \leq n)\}$ and $glb(\Sigma, A) = \sup\{n : \Sigma \approx\!\!| (A \geq n)\}$, respectively.

3 Some basic properties

We recall some salient properties of the logic $\mathcal{L}^M$. For any proposition A, B and C, it is easy to verify that

$$\begin{aligned} \neg\top &\equiv \bot && (11) \\ A \wedge \top &\equiv A && (12) \\ A \vee \top &\equiv \top && (13) \\ A \wedge \bot &\equiv \bot && (14) \\ A \vee \bot &\equiv A && (15) \\ \neg\neg A &\equiv A && (16) \\ \neg(A \wedge B) &\equiv \neg A \vee \neg B && (17) \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B && (18) \\ A \rightarrow B &\equiv \neg A \vee B && (19) \\ (A \wedge (B \vee C)) &\equiv (A \wedge B) \vee (A \wedge C) && (20) \\ (A \vee (B \wedge C)) &\equiv (A \vee B) \wedge (a \vee C) && (21) \end{aligned}$$

It can be verified that each proposition may easily be transformed, by preserving equivalence, into either $\top$, $\bot$ or a proposition in NNF, CNF and DNF in which neither $\top$ nor $\bot$ occur. Please note, we do not have $A \wedge \neg A \equiv \bot$. In general we can only say that $(A \wedge \neg A)^{\mathcal{I}} \leq 0.5$, for any interpretation $\mathcal{I}$ and similarly $(A \vee \neg A)^{\mathcal{I}} \geq 0.5$.

Concerning, meta propositions we have the equivalencies of classical propositional logic as meta propositions have a boolean truth value, e.g. $\psi \wedge \top \equiv \psi$, $\neg\psi \wedge \psi \equiv \bot$, as well as

$$(\top \geq n) \equiv \top \quad (22)$$

$$(\top \leq n) \equiv \begin{cases} \top & \text{if } n = 1 \\ \bot & \text{otherwise} \end{cases} \quad (23)$$

$$(\neg A \geq n) \equiv (A \leq 1 - n) \quad (24)$$

$$(A \wedge B \geq n) \equiv (A \geq n) \wedge (B \geq n) \quad (25)$$

$$(A \vee B \geq n) \equiv (A \geq n) \vee (B \geq n) \quad (26)$$

and likewise for the cases $\leq, <$ and $>$. Therefore, each meta proposition may easily be transformed by, preserving equivalence, into $\top$, $\bot$ or into a meta proposition in NNF, CNF and DNF in which neither $\top$ nor $\bot$ occur.

Since $\Sigma \approx \top, \Sigma \not\approx \bot, glb(\Sigma, \top) = 1, glb(\Sigma, \bot) = 0, lub(\Sigma, \top) = 1, lub(\Sigma, \bot) = 0$, $\Sigma \cup \{\top\}$ and Σ share the same set of models, and that $\Sigma \cup \{\bot\}$ is unsatisfiable, for the rest of the paper, if not stated otherwise, *we will always assume that meta propositions are always in NNF, non-trivial and neither* $\top$ *nor* $\bot$ *occur* in them.

There is a strict relation between meta propositions and classical propositions. Let us consider the following transformation ${}^c(\cdot)$ of meta propositions into propositions, where ${}^c(\cdot)$ takes the "crisp" propositional part of meta propositions:

$$ {}^c(p \geq n) \mapsto p \quad (27)$$

$$ {}^c(p \leq n) \mapsto \neg p \quad (28)$$

$$ {}^c(\neg \psi) \mapsto \neg {}^c\psi \quad (29)$$

$$ {}^c(\psi_1 \wedge \psi_2) \mapsto {}^c\psi_1 \wedge {}^c\psi_2 \quad (30)$$

$$ {}^c(\psi_1 \vee \psi_2) \mapsto {}^c\psi_1 \vee {}^c\psi_2 \quad (31)$$

$\psi_1 \to \psi_2$ is handled via $\neg\psi_1 \vee \psi_2$ and, for a meta theory Σ, ${}^c\Sigma = \{{}^c\psi : \psi \in \Sigma\}$.

Proposition 1 *For a meta theory Σ, if Σ is unsatisfiable then ${}^c\Sigma$ is classically unsatisfiable.* $\dashv$

Proof: Assume Σ unsatisfiable. Now, suppose to the contrary that ${}^c\Sigma$ is satisfiable. Therefore there is a classical propositional model ${}^c\mathcal{I}$ of ${}^c\Sigma$. Let $\mathcal{I}$ be the following interpretation. For all propositional letters p,

$$p^{\mathcal{I}} = \begin{cases} 1 & \text{if } {}^c\mathcal{I} \text{ satisfies } p \\ 0 & \text{otherwise} \end{cases}$$

Now, we will show that $\mathcal{I}$ is an interpretation satisfying Σ which is contrary to our assumption. At first, we show on induction on the number of connectives of ψ that ${}^c\mathcal{I}$ satisfies ${}^c\psi$ iff $\mathcal{I}$ satisfies ψ.

ψ is a meta letter:

1. Suppose ψ is a meta letter $(p \geq n)$. Therefore, ${}^c\psi = p$. If ${}^c\mathcal{I}$ satisfies p then, by definition, $p^{\mathcal{I}} = 1$ and, thus, $\mathcal{I}$ satisfies $(p \geq n)$. If ${}^c\mathcal{I}$ does not satisfy p then, by definition, $p^{\mathcal{I}} = 0$ and, thus, $\mathcal{I}$ does not satisfy $(p \geq n)$ (note that $n > 0$).
2. Suppose ψ is a meta letter $(p \leq n)$. Therefore, ${}^c\psi = \neg p$. If ${}^c\mathcal{I}$ satisfies $\neg p$ then, by definition, $p^{\mathcal{I}} = 0$ and, thus, $\mathcal{I}$ satisfies $(p \leq n)$. If ${}^c\mathcal{I}$ does not satisfy $\neg p$ then, by definition, $p^{\mathcal{I}} = 1$ and, thus, $\mathcal{I}$ does not satisfy $(p \leq n)$ (note that $n < 1$).

Induction step:

1. Suppose ψ is a meta proposition $\neg\psi'$. Therefore, ${}^c\psi = \neg {}^c\psi'$. If ${}^c\mathcal{I}$ satisfies $\neg {}^c\psi'$ then, by induction on ${}^c\psi'$, $\mathcal{I}$ does not satisfy ψ' and, thus, $\mathcal{I}$ satisfies ψ. If ${}^c\mathcal{I}$ does not satisfy $\neg {}^c\psi'$ then, by induction on ${}^c\psi'$, $\mathcal{I}$ satisfies ψ' and, thus, $\mathcal{I}$ does not satisfy ψ.

2. Suppose ψ is a meta proposition $\psi_1 \wedge \psi_2$. Therefore, ${}^c\psi = {}^c\psi_1 \wedge {}^c\psi_2$. If ${}^c\mathcal{I}$ satisfies ${}^c\psi$ then, ${}^c\mathcal{I}$ satisfies both ${}^c\psi_1$ and ${}^c\psi_2$. By induction on ${}^c\psi_1$ and ${}^c\psi_2$, $\mathcal{I}$ satisfies ψ_1 and ψ_2 and, thus, $\mathcal{I}$ satisfies ψ. If ${}^c\mathcal{I}$ does not satisfy ${}^c\psi$ then, ${}^c\mathcal{I}$ either does not satisfy ${}^c\psi_1$ or does not satisfy ${}^c\psi_2$. By induction on ${}^c\psi_1$ and ${}^c\psi_2$, $\mathcal{I}$ either does not satisfy ψ_1 or does not satisfy ψ_2 and, thus, $\mathcal{I}$ does not satisfy ψ.
3. The cases $\vee$ is similar.

Therefore, for any $\psi \in \Sigma$, since both ${}^c\psi \in {}^c\Sigma$ and ${}^c\mathcal{I}$ satisfies ${}^c\psi$, it follows that $\mathcal{I}$ satisfies any $\psi \in \Sigma$ and, thus, $\mathcal{I}$ satisfies Σ which is contrary to our assumption. `Q.E.D.`

Corollary 1 *Let Σ be a meta theory and let ψ be a meta proposition. If $\Sigma {\models\!\!\approx} \psi$ then ${}^c\Sigma \models {}^c\psi$, where $\models$ is classical entailment.* $\dashv$

Proof: If $\Sigma {\models\!\!\approx} \psi$ then $\Sigma \cup \{\neg\psi\}$ is not satisfiable and, thus, by Proposition 1, ${}^c\Sigma \cup \{\neg {}^c\psi\}$ is not classically satisfiable. Therefore, ${}^c\Sigma \models {}^c\psi$ holds. `Q.E.D.`

Proposition 1 states that *there cannot be entailment without classical entailment.* In this sense ${\models\!\!\approx}$ is correct with respect to $\models$.

Example 1 Let Σ be the set $\Sigma = \{(p{\geq}0.6) \vee (q{\leq}0.3), (p{\leq}0.3)\}$. Let ψ be $(q{\leq}0.8)$. It follows that ${}^c\Sigma = \{p \vee \neg q, \neg p\}$. It is easily verified that $\Sigma {\models\!\!\approx} (q{\leq}0.8)$ and that ${}^c\Sigma \models \neg q$, thereby confirming Proposition 1. ■

The converse of Corollary 1 does not hold in the general case.

Example 2 Let Σ be the set $\Sigma = \{(p{\leq}0.6) \vee (q{\geq}0.7), (p{\geq}0.3)\}$. It follows that ${}^c\Sigma = \{\neg p \vee q, p\}$. It is easily verified that ${}^c\Sigma \models q$, but $\Sigma {\not\models\!\!\approx} (q{\geq}n)$, for all $n > 0$. ■

In Section 4.1 we will present a result which establishes the converse of Corollary 1.

As it happens for classical entailment, entailment in $\mathcal{L}^M$ can be reduced to satisfiability checking: indeed, for a meta theory Σ and a meta proposition ψ

$$\Sigma {\models\!\!\approx} \psi \quad \text{iff} \quad \Sigma \cup \{\neg\psi\} \text{ is not satisfiable} \tag{32}$$

holds.

We conclude this section by showing that the computation of the least upper bound can be reduced to the computation of the greatest lower bound. Let Σ be a meta theory and let A be a proposition. By (24), $(A{\leq}n) \equiv (\neg A{\geq}1-n)$ holds and, thus,

$$\Sigma {\models\!\!\approx} (A{\leq}n) \quad \text{iff} \quad \Sigma {\models\!\!\approx} (\neg A{\geq}1-n) \tag{33}$$

Therefore,

$$\begin{array}{rcl} 1 - lub(\Sigma, A) & = & 1 - \inf\{n : \Sigma {\models\!\!\approx} (A{\leq}n)\} \\ & = & \sup\{1-n : \Sigma {\models\!\!\approx} (A{\leq}n)\} \\ & = & \sup\{n : \Sigma {\models\!\!\approx} (A{\leq}1-n)\} \\ & = & \sup\{n : \Sigma {\models\!\!\approx} (\neg A{\geq}n)\} \\ & = & glb(\Sigma, \neg A). \end{array}$$

and, thus,

$$lub(\Sigma, A) \quad = \quad 1 - glb(\Sigma, \neg A), \tag{34}$$

i.e. the *lub* can be determined through the *glb* (and vice-versa).

4 Decision algorithms in $\mathcal{L}^M$

In the following two sections we will present algorithms for deciding the two main problems within $\mathcal{L}^M$: (*i*) first we describe a decision algorithm for deciding the satisfiability problem in $\mathcal{L}^M$, i.e. deciding whether a meta theory Σ is satisfiable or not (by (32), the entailment problem is solved too). We call it the *fuzzy SAT problem* in order to distinguish it from the classical SAT problem; and (*ii*) we present two algorithms for deciding the BTVB problem, i.e. we provide two algorithms which determine the greatest lower bound of a proposition A with respect to a meta theory Σ (thus, by (34), the least upper bound can be determined, too).

4.1 A decision algorithm for the fuzzy SAT problem

In [1] both an axiom system for deciding entailment as well as a resolution based method for the fuzzy SAT problem have been defined. In this latter case, the transformation of meta propositions into an equivalent CNFs is required. We propose a simple alternative (a semantic tableaux, see e.g. [2, 5]) which is a variant of the classical DPLL (Davis-Putnam-Longemann-Loveland) procedure [3, 4] for classical propositional logic not in conjunctive normal form (no CNF conversions is needed, which may require exponential time).

Given two meta propositions ψ_1 and ψ_2 we say that (*i*) ψ_1 *subsumes* ψ_2 (denoted by $subs(\psi_1, \psi_2)$) iff $\psi_1 \approx\!\!\!\models \psi_2$; and that (*ii*) ψ_1 and ψ_2 are *pairwise contradictory* (denoted by $ctd(\psi_1, \psi_2)$) iff $\psi_1 \approx\!\!\!\models \neg\psi_2$. For instance, $(p{\geq}0.3) \vee (q{\leq}0.6)$ subsumes $(p{\geq}0.4) \vee (q{\leq}0.9)$, while $(p{\geq}0.3) \vee (q{\leq}0.6)$ and $(p{\leq}0.2) \wedge (q{\geq}0.7)$ are pairwise contradictory. Since $\psi_1 \approx\!\!\!\models \neg\psi_2$ iff $\psi_2 \approx\!\!\!\models \neg\psi_1$ it follows that $ctd(\cdot,\cdot)$ is symmetric. By definition,

$$ctd(\psi_1, \psi_2) \quad \text{iff} \quad subs(\psi_1, \neg\psi_2) \tag{35}$$

holds which relates $ctd(\cdot,\cdot)$ to $subs(\cdot,\cdot)$. If ψ_1 and ψ_2 are two meta literal, it is quite easy to check whether $subs(\psi_1, \psi_2)$ holds, as shown in Table 1, on the left. Each entry in the table specifies the condition under which ψ_1 subsumes ψ_2.

ψ_1	ψ_2			
	$(p{\geq}m)$	$(p{>}m)$	$(p{\leq}m)$	$(p{<}m)$
$(p{\geq}n)$	$n \geq m$	$n > m$	$\times$	$\times$
$(p{>}n)$	$n \geq m$	$n \geq m$	$\times$	$\times$
$(p{\leq}n)$	$\times$	$\times$	$n \leq m$	$n < m$
$(p{<}n)$	$\times$	$\times$	$n \leq m$	$n \leq m$

ψ_1	ψ_2	
	$(p{\geq}m)$	$(p{>}m)$
$(p{\leq}n)$	$n < m$	$n \leq m$
$(p{<}n)$	$n \leq m$	$n \leq m$

Table 1: On the left: ψ_1 subsumes ψ_2. On the right: ψ_1 and ψ_2 pairwise contradictory.

We are now ready to specify the calculus. Let Σ be a meta theory. The calculus is based on the following rules described in Table 2.
In order to prevent infinite application of the above rules, we assume that each instantiation of the rules is applied only once.

We define $\mathcal{R}^{DPLL} = \{(up), (\wedge), (\vee)\}$. Note that the rule in $\mathcal{R}^{DPLL}$ form a variant of the classical DPLL (Davis-Putnam-Longemann-Loveland) procedure [3, 4] for classical propositional logic not in conjunctive normal form: the (*up*) rule is the usual *unit propagation* rule, whereas the $(\vee)$ rule (the only branching rule) is also called *Principle of Bivalence* [2] and extends the usual tableaux $(\vee)$ rule

$$(\vee_s) \quad \frac{\psi_1 \vee \psi_2}{\psi_1 \mid \psi_2} \tag{36}$$

$$(up)\ \frac{\psi, \psi_l}{\psi'}$$ where ψ_l is a literal and ψ' is obtained from ψ by replacing each occurrence of a literal ψ'' in ψ by (i) $\top$, if $subs(\psi_l, \psi'')$; and (ii) $\bot$, if $ctd(\psi_l, \psi'')$, followed by boolean simplification

$$(\wedge)\ \frac{\psi_1 \wedge \psi_2}{\psi_1, \psi_2}$$

$$(\vee)\ \frac{\psi_1 \vee \psi_2}{\psi_1 \mid \psi', \psi_2}$$ where ψ' is a NNF of $\neg\psi_1$

Table 2: Semantic tableaux inference rules with unit propagation.

Further, it is worth mentioning that the resolution rule for meta propositions in CNF presented in [1] can be defined as[2]

$$(res)\ \frac{\psi_{l_1} \vee \psi, \psi_{l_2} \vee \psi'}{\psi \vee \psi'} \quad \text{where } \psi_{l_1}\ \psi_{l_2} \text{ are two meta literal and } ctd(\psi_{l_1}, \psi_{l_2}) \tag{37}$$

As usual, a deduction is represented as a tree, called *deduction tree*. A branch ϕ in a deduction tree is closed iff it contains $\bot$. A deduction tree is *closed* iff each branch in it is closed. With ϕ^M we indicate the set of meta propositions occurring in ϕ. A meta theory Σ has a *refutation* iff each deduction tree is closed. A branch ϕ is *completed* iff it is not closed and no rule can be further applied to it. A branch ϕ is *open* iff it is not closed and not completed.

The algorithm $\mathsf{SAT}(\Sigma)$ described in Table 3 determines whether Σ is satisfiable or not. $\mathsf{SAT}(\Sigma)$ starts from the root labelled Σ and applies the rules until the resulting tree is either closed or there is a completed branch. If the tree is closed, $\mathsf{SAT}(\Sigma)$ returns false, otherwise true and from the completed branch a model of Σ can be build. The set of not closed branches ϕ which may be expanded during the deduction is hold by Φ. If Φ is managed as a stack then we have a depth first search strategy, while if Φ is managed as a queue then a breath first search strategy is applied.

Example 3 Let Σ be the set

$$\begin{aligned}\Sigma \;=\; & \{(p{\geq}0.5) \vee ((q{\geq}0.4) \wedge (u{\geq}0.6)), (p{\leq}0.2) \vee (q{\geq}0.6),\\ & (q{\leq}0.5) \vee (r{\geq}0.4) \vee (s{\geq}0.5), (r{\leq}0.3) \vee (t{\geq}0.6),\\ & (r{\leq}0.2) \vee (t{\leq}0.5), (r{\geq}0.7) \vee (s{\leq}0.3), (u{\leq}0.5) \vee (r{\leq}0.1)\}\end{aligned}$$

Figure 1 shows a deduction tree produced by $\mathsf{SAT}(\Sigma)$. The two branches on the left are closed, while the branch ϕ on the right is completed. Consider $\phi'^M \subseteq \phi^M$ where ϕ'^M contains all the meta literal occurring in ϕ^M, i.e.

$$\begin{aligned}\phi'^M \;=\; & \{(p{<}0.5), (p{\leq}0.2), (q{\leq}0.4),\\ & (u{\geq}0.6), (s{\leq}0.3), (r{\leq}0.1)\}\end{aligned}$$

From ϕ'^M a model $\mathcal{I}$ of Σ can easily be build as follows: $p^{\mathcal{I}} = 0.2, q^{\mathcal{I}} = 0.4, u^{\mathcal{I}} = 0.6, s^{\mathcal{I}} = 0.3, r^{\mathcal{I}} = 0.1$ and $t^{\mathcal{I}} = 0$. ■

The above example directly suggests us how to prove the following proposition which establishes correctness and completeness of the SAT algorithm.

Proposition 2 *Let Σ be a meta theory. Then $\mathsf{SAT}(\Sigma)$ iff Σ is satisfiable.* ⊣

[2]$\vee$ is a symmetric relation.

Algorithm $\mathsf{SAT}(\Sigma)$**:**

$\mathsf{SAT}(\Sigma)$ starts from the root labelled Σ. So, we initialise Φ with $\Phi = \{\phi\}$, where $\phi^M = \Sigma$. Φ is managed as a multiset, i.e. there could be elements in Φ which are replicated.

1. if $\Phi = \emptyset$ then return `false` and exit;

   `/*` all branches are closed, thus, Σ is unsatisfiable `*/`

2. otherwise, select a branch $\phi \in \Phi$ and remove it from Φ, i.e. $\Phi \leftarrow \Phi \setminus \{\phi\}$;

3. try to apply a rule to ϕ with the following priority among the rules: $(up) \succ (\wedge) \succ (\vee)$:

   (a) if the (up) rule is applicable to ϕ then expand ϕ by repeated application of the (up) rule until (up) is no more applicable. Let ϕ' be the resulting branch. If ϕ' is not closed then add it to Φ, i.e. $\Phi \leftarrow \Phi \cup \{\phi'\}$. Go to step 1.

   (b) if the $(\wedge)$ rule is applicable to ϕ then expand ϕ by repeated application of the $(\wedge)$ rule until $(\wedge)$ is no more applicable. Let ϕ' be the resulting branch. If ϕ' is not closed then add it to Φ, i.e. $\Phi \leftarrow \Phi \cup \{\phi'\}$; Go to step 1.

   (c) if the $(\vee)$ rule is applicable to ϕ then expand ϕ by *one* application of the $(\vee)$ rule. Let ϕ_1 and ϕ_2 be the resulting branches. For each $\phi_i, i = 1, 2$, if ϕ_i is not closed then add it to Φ, i.e. $\Phi \leftarrow \Phi \cup \{\phi_i\}$. Go to step 1.

   (d) otherwise, if no rule is applicable to ϕ, then return `true` and exit.

   `/*` ϕ is completed, thus, Σ is satisfiable `*/`

Table 3: The SAT procedure.

Proof: It can be easily verified that the rules in $\mathcal{R}^{DPLL}$ are correct, i.e. for a branch ϕ, ϕ^M is satisfiable iff there is a branch ϕ' as the result of the application of a rule to ϕ such that ϕ'^M satisfiable.

$\Rightarrow$.) Suppose $\mathsf{SAT}(\Sigma)$. Let T be the generated deduction tree and let ϕ be a completed branch from Σ to a leaf in T. Such a branch has to exist, otherwise $\mathsf{SAT}(\Sigma) =$ `false`. For any letter p, let $(\max\emptyset = 0, \min\emptyset = 1)$:

$$glb_p^{\geq} = \max\{n : (p \geq n) \in \phi^M\} \tag{38}$$

$$glb_p^{>} = \max\{n : (p > n) \in \phi^M\} \tag{39}$$

$$lub_p^{\leq} = \min\{n : (p \leq n) \in \phi^M\} \tag{40}$$

$$lub_p^{<} = \min\{n : (p < n) \in \phi^M\} \tag{41}$$

For any p, $glb_p^{\geq}, glb_p^{>}$ and $lub_p^{\leq}, lub_p^{<}$, determine the greatest lower bound and the least upper bound which p's truth value has to satisfy. Since ϕ is not closed, it follows that for each letter p, there is $\epsilon_p \geq 0$ such that

$$glb_p = \max\{glb_p^{\geq}, glb_p^{>} + \epsilon_p\} \leq \min\{lub_p^{\leq}, lub_p^{<} - \epsilon_p\} = lub_p \tag{42}$$

i.e., for each p, its greatest lower bound constraint is less or equal than its least upper bound constraint. Now, let $\mathcal{I}$ be an interpretation such that

1. $\top^{\mathcal{I}} = 1$ and $\bot^{\mathcal{I}} = 0$;

2. $p^{\mathcal{I}} = glb_p$, for all letters p.

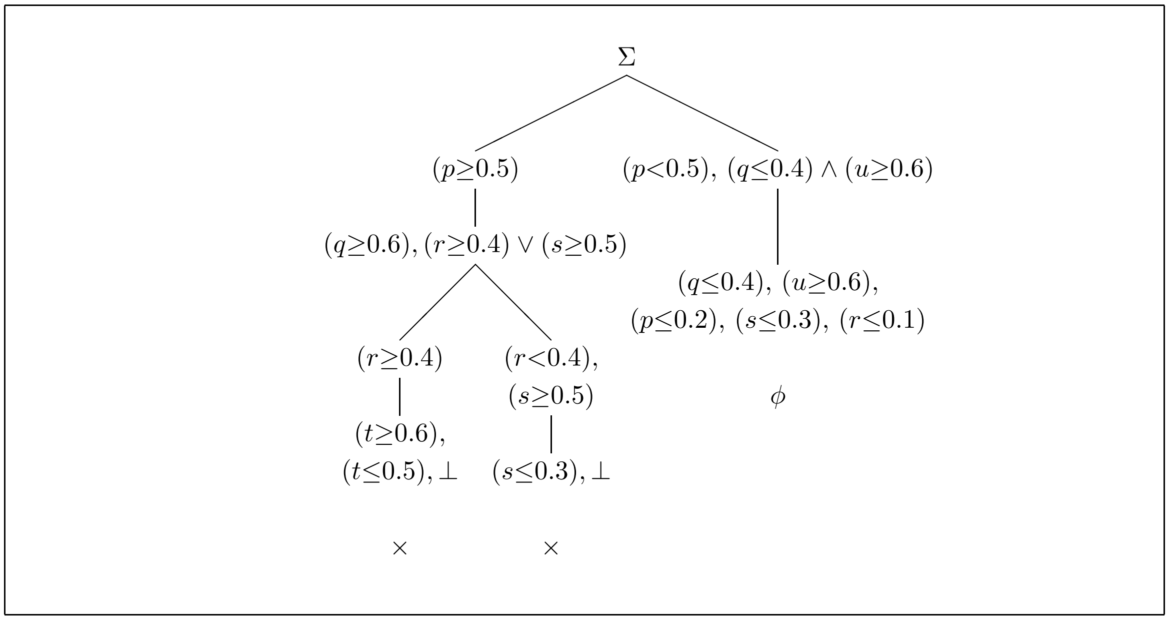

Σ
$(p{\geq}0.5)$
$(p{<}0.5)$, $(q{\leq}0.4) \wedge (u{\geq}0.6)$
$(q{\geq}0.6), (r{\geq}0.4) \vee (s{\geq}0.5)$
$(q{\leq}0.4)$, $(u{\geq}0.6)$, $(p{\leq}0.2)$, $(s{\leq}0.3)$, $(r{\leq}0.1)$
$(r{\geq}0.4)$
$(r{<}0.4)$, $(s{\geq}0.5)$
ϕ
$(t{\geq}0.6)$, $(t{\leq}0.5), \perp$
$(s{\leq}0.3), \perp$
$\times$
$\times$

Figure 1: Deduction tree for Σ.

It is easily verified that $\mathcal{I}$ satisfies all $\psi \in \phi^M$ (by induction on the number of connectives in ψ and since ϕ is completed) and, thus, $\mathcal{I}$ satisfies ϕ^M. As a consequence, $\Sigma \subseteq \phi^M$ is satisfiable.
$\Leftarrow$.) Suppose Σ is satisfiable. Let T be the generated deduction tree. From the correctness of the rules it follows that there is a branch ϕ in T such that (i) ϕ^M is satisfiable; (ii) no rule can be further applied to ϕ; and (iii) ϕ is not closed. Therefore, ϕ is completed and, thus, $\mathsf{SAT}(\Sigma)$.

Q.E.D.

The following result establishes the converse of Proposition 1 in case we restrict our attention to meta propositions in NNF involving truth values above 0.5. It directly relates to a similar result described in [9]. At first, we replace the (up) rule in SAT with a simplified version of it:

$$(\perp)\ \frac{\psi_{l_1}, \psi_{l_2}}{\perp} \quad \text{where } \psi_{l_1}\ \psi_{l_2} \text{ are two meta literal and } ctd(\psi_{l_1}, \psi_{l_2}) \tag{43}$$

It is easily verified that $\mathcal{R}^T = \{(\perp), (\wedge), (\vee)\}$ forms a correct and complete set of rules for which Proposition 2 still holds[3]. Let ${}^c\mathcal{R}^T = \{{}^c(\perp), ({}^c(\wedge), {}^c(\vee)\}$ be the classical rules of inference analogue to the $(\perp), (\wedge)$ and $(\vee)$ and let ${}^c\mathsf{SAT}$ be the SAT procedure where the inference rules $(\perp), (\wedge)$ and $(\vee)$ have been replaced with ${}^c(\perp), {}^c(\wedge)$ and ${}^c(\vee)$. It is well known that ${}^c\mathsf{SAT}$ is correct and complete, i.e. for any set of classical propositions Σ, Σ is classically satisfiable iff ${}^c\mathsf{SAT}(\Sigma)$ returns `true`. Finally, we will say that a meta proposition ψ is *normalised* iff for each meta literal ψ' occurring in ψ,

1. if ψ' is $(p{\geq}n)$ then $n > 0.5$;
2. if ψ' is $(p{\leq}n)$ then $n < 0.5$;
3. if ψ' is $(p{>}n)$ then $n \geq 0.5$;
4. if ψ' is $(p{<}n)$ then $n \leq 0.5$.

[3]The same holds by considering the rules $\mathcal{R}^R = \{(\perp), (res)\}$ only, if meta propositions are in CNF.

The following proposition holds.

Proposition 3 *Let Σ be a meta theory such that each $\psi \in \Sigma$ is normalised. Then* $\mathsf{SAT}(\Sigma)$ *iff* ${}^c\mathsf{SAT}({}^c\Sigma)$. $\dashv$

Proof:
$\Rightarrow$.) Assume $\mathsf{SAT}(\Sigma)$ and let T be the deduction tree build by $\mathsf{SAT}(\Sigma)$. Let ϕ be a branch in T. We show by induction on the depth $depth(\phi)$ of ϕ that there is branch ${}^c\phi$ in a deduction tree ${}^cT$ build by ${}^c\mathsf{SAT}({}^c\Sigma)$ such that ${}^c(\phi^M) = ({}^c\phi)^M$.

$depth(\phi) = 0$**:** Then $\phi^M = \Sigma$ and ${}^c(\phi^M) = {}^c\Sigma$. So, just consider the initial branch ${}^c\phi$ of a ${}^c\mathsf{SAT}({}^c\Sigma)$ deduction. By definition of ${}^c\mathsf{SAT}$, $({}^c\phi)^M = {}^c\Sigma$ and, thus, ${}^c(\phi^M) = ({}^c\phi)^M$.

$depth(\phi) = n > 0$**:** Suppose that for each branch ϕ' such that $depth(\phi') < n$, the above induction hypothesis holds. ϕ is the result of the application of one of the $(\bot)$, $(\wedge)$ and $(\vee)$ rules the branch ϕ' of depth $depth(\phi') = n-1$. By induction on ϕ', let ${}^c\phi'$ be a branch in a deduction tree ${}^cT$ build by ${}^c\mathsf{SAT}({}^c\Sigma)$ such that ${}^c(\phi'^M) = ({}^c\phi')^M$.

(i) If the $(\bot)$ rule has been applied to ϕ' then $\psi_{l_1}, \psi_{l_2} \in \phi'^M$ such that $ctd(\psi_{l_1}, \psi_{l_2})$ and $\bot \in \phi$. Therefore, ${}^c\psi_{l_1}, {}^c\psi_{l_2} \in {}^c\phi'^M$, $ctd({}^c\psi_{l_1}, {}^c\psi_{l_2})$, the $({}^c\bot)$ can be applied to ${}^c\phi'$ and, thus, $\bot \in {}^c\phi$. Therefore, ${}^c(\phi^M) = ({}^c\phi)^M$.

(ii) The cases for the $(\wedge)$ rule or $(\vee)$ rule can be shown similarly.

Since $\mathsf{SAT}(\Sigma)$, there is a completed branch ϕ and, thus, there is a completed branch ${}^c\phi$ in a deduction tree ${}^cT$ build by ${}^c\mathsf{SAT}({}^c\Sigma)$. Therefore, ${}^c\mathsf{SAT}({}^c\Sigma)$ returns `true`.
$\Leftarrow$.) Assume ${}^c\mathsf{SAT}({}^c\Sigma)$ and let ${}^cT$ be the deduction tree build by ${}^c\mathsf{SAT}({}^c\Sigma)$. Let ${}^c\phi$ be a branch in ${}^cT$. We show by induction on the depth $depth({}^c\phi)$ of ${}^c\phi$ that there is branch ϕ in a deduction tree T build by $\mathsf{SAT}(\Sigma)$ such that $({}^c\phi)^M = {}^c(\phi^M)$.

$depth({}^c\phi) = 0$**:** Similarly as above.

$depth({}^c\phi) = n > 0$**:** Suppose that for each branch ${}^c\phi'$ such that $depth({}^c\phi') < n$, the above induction hypothesis holds. ${}^c\phi$ is the result of the application of one of the ${}^c(\bot)$, ${}^c(\wedge)$ and ${}^c(\vee)$ rules the branch ${}^c\phi'$ of depth $depth({}^c\phi') = n-1$. By induction on ${}^c\phi'$, let ϕ' be a branch in a deduction tree T build by $\mathsf{SAT}(\Sigma)$ such that $({}^c\phi')^M = {}^c(\phi'^M)$.

(i) Suppose the ${}^c(\bot)$ rule has been applied to ${}^c\phi'$ and let $p, q \in ({}^c\phi')^M$ be the literal to which the rule has been applied. Therefore, $ctd(p, q)$ and $\bot \in {}^c\phi$. Since ${}^c\phi'^M = {}^c(\phi'^M)$ holds, there are two meta literals $\psi_{l_1}, \psi_{l_2} \in \phi'^M$ such that ${}^c\psi_{l_1} = p$ and ${}^c\psi_{l_2} = q$. Suppose ${}^c\psi_{l_1}$ is $(p \geq n)$ and ${}^c\psi_{l_2}$ is $(p \leq m)$. Since, by hypothesis, $n > 0.5$ and $m < 0.5$, it follows that $ctd(\psi_{l_1}, \psi_{l_2})$, the $(\bot)$ can be applied to ϕ' and, thus, $\bot \in \phi$. Therefore, $({}^c\phi)^M = {}^c(\phi^M)$. The other cases for $>, \leq$ and $<$ are similar.

(ii) The cases for the $({}^c\wedge)$ rule or $({}^c\vee)$ rule can be proven similarly.

Since ${}^c\mathsf{SAT}({}^c\Sigma)$, there is a completed branch ${}^c\phi$ and, thus, $({}^c\phi)^M$ is satisfiable. Therefore, there is a branch ϕ in a deduction tree T build by $\mathsf{SAT}(\Sigma)$ such that $({}^c\phi)^M = {}^c(\phi^M)$. If ϕ is completed, then $\mathsf{SAT}(\Sigma)$ returns `true`. Otherwise, ϕ can be expanded to a completed branch ϕ' (else, by $\Rightarrow$.) each expanded branch of ${}^c\phi$ is closed and, thus, ${}^c(\phi^M)$ is not satisfiable). Therefore, $\mathsf{SAT}(\Sigma)$ returns `true`. `Q.E.D.`

Essentially, the above proposition establishes that once the truth value of a meta letter ψ is restricted to be above (or below) the threshold 0.5, then the proposition ${}^c\psi$ may definitely be considered as classically true (false, respectively).

Corollary 2 *Let Σ be a meta theory and let ψ be a meta proposition. Furthermore, we assume that each $\psi' \in \Sigma$ is normalised as well as any equivalent NNF of $\neg\psi$ is normalised. Then*

1. *Σ is satisfiable iff ${}^c\Sigma$ is classically satisfiable;*

2. *$\Sigma \approx\!\!\!\!\models \psi$ iff ${}^c\Sigma \models {}^c\psi$, where $\models$ is classical entailment.* $\dashv$

Example 4 Consider Example 1. An equivalent NNF of $\neg\psi$ is $\psi' = (q{>}0.8)$. It is easily verified that both Σ and ψ are normalised. Indeed, both $\Sigma \approx\!\!\!\!\models \psi$ and ${}^c\Sigma \models {}^c\psi$ hold. On the other hand, in Example 2, Σ is not normalised, e.g. for $(p{\geq}0.3)$ we have $0.3 < 0.5$.

■

From a computational point of view, the fuzzy SAT problem and the classical (NP-complete) SAT problem are in the same complexity class.

Proposition 4 *Let Σ be a meta theory. Then checking whether Σ is satisfiable is a NP-complete problem.* $\dashv$

Proof: Consider a crisp propositional theory K in NNF such that neither $\bot$ nor $\top$ occur in K. We define $\Sigma_K = \{\psi : \psi$ is a NNF of $(A{\geq}1)$ and $A \in K\}$. Since ${}^c\Sigma_K$ and K have the same classical models and since Σ_K satisfies the conditions of Corollary 2, it follows that K is classically satisfiable iff Σ_K is satisfiable. As a consequence, the NP-hardness of the fuzzy SAT problem follows. The following NP algorithm determines whether Σ is satisfiable. Non-deterministically generate a complete branch ϕ in a deduction tree for Σ: Σ is satisfiable iff such ϕ exists. The depth of ϕ is polynomially bounded by the input. Hence, deciding whether Σ is satisfiable is in NP. `Q.E.D.`

From the coNP-completeness of the classical entailment problem, it follows similarly

Corollary 3 *Let Σ be a meta theory and let ψ be a meta proposition. Then checking whether $\Sigma \approx\!\!\!\!\models \psi$ is a coNP-complete problem.* $\dashv$

4.2 A decision algorithm for the BTVB problem

We address now the problem of determining $glb(\Sigma, A)$ and $lub(\Sigma, A)$. This is important, as computing, e.g. $glb(\Sigma, A)$, is in fact the way to answer a query of type "to which degree is A (at least) true, given the (imprecise) facts in Σ ?". Since $lub(\cdot)$ can be defined in terms of $glb(\cdot)$ (see Equation (34)), we will restrict our attention to this latter case only.

In the following let $\Phi_\Sigma = \{\phi_1, \ldots, \phi_k\}$ be the set of all completed branches of a deduction tree computed by the algorithm $\mathsf{COMPL}(\Sigma)$, where $\mathsf{COMPL}(\Sigma)$ is as $\mathsf{SAT}(\Sigma)$ except that Point 1. and Point 3d have been replaced with

- "if $\Phi = \emptyset$ then return Φ_Σ and exit";
- "otherwise, if no rule is applicable to ϕ, then add ϕ to Φ_Σ. Go to step 1".

respectively. Of course, if $\Phi_\Sigma = \emptyset$ then Σ is unsatisfiable and, thus, $glb(\Sigma, A) = 1$. Furthermore, for each branch ϕ, let

$$\begin{array}{rcll}\phi^A & = & \{\psi \in \phi^M : & \psi \text{ is a meta literal and} \\ & & & \text{there is no meta literal } \psi' \in \phi^M \\ & & & \text{such that } \psi \neq \psi' \text{ and } subs(\psi', \psi)\}\end{array} \qquad (44)$$

It follows that for all letters p, for each completed branch $\phi \in \Phi^\Sigma$, $\phi^A \subseteq \phi^M$ contains the most informative truth value constrains for p, i.e. there are at most two expressions of the form $(p\ r\ n)$ and $(p\ r'\ m)$ in ϕ^M such that $r \in \{\geq, >\}$ and $r' \in \{\leq, <\}$. For instance, in case of the completed branch ϕ in Figure 1 (see Example 3), ϕ^A is $\{(p{\leq}0.2), (q{\leq}0.4), (u{\geq}0.6), (s{\leq}0.3), (r{\leq}0.1)\}$. For any letter p and for each branch ϕ, let us define

$$
\begin{array}{rcl}
pos(p,\phi) & = & \left\{ \begin{array}{ll} n & \text{if } (p{\geq}n) \in \phi^A \text{ or } (p{>}n) \in \phi^A \\ 0 & \text{otherwise} \end{array} \right. \\
neg(p,\phi) & = & \left\{ \begin{array}{ll} 1-n & \text{if } (p{\leq}n) \in \phi^A \text{ or } (p{<}n) \in \phi^A \\ 0 & \text{otherwise} \end{array} \right.
\end{array}
\tag{45}
$$

Essentially, $pos(p, \phi)$ and $neg(p, \phi)$ express the least truth value constrain of p with respect to ϕ and the least truth value constrain of $\neg p$ with respect to ϕ, respectively. For instance, for Example 3 we have that e.g. $pos(p, \phi) = 0$ and $neg(p, \phi) = 0.8$ while $pos(u, \phi) = 0.6$ and $neg(u, \phi) = 0$ hold. We extend this notion to an arbitrary proposition A in NNF as follows. The *truth value of* A with respect to a branch ϕ is defined inductively as follows:

$$
val(A,\phi) = \left\{ \begin{array}{ll} pos(p,\phi) & \text{if } A \text{ is a letter } p \\ neg(p,\phi) & \text{if } A \text{ is a literal } \neg p \\ \min\{val(A_1,\phi), val(A_2,\phi) & \text{if } A \text{ is a } A_1 \wedge A_2 \\ \max\{val(A_1,\phi), val(A_2,\phi) & \text{if } A \text{ is a } A_1 \vee A_2 \end{array} \right.
\tag{46}
$$

For instance, with respect to Example 3 and $A = \neg p \wedge u$, we have $val(A, \phi) = 0.6$. Further, it is quite easily verified that for any proposition A, A' such that $A \equiv A'$ and branch ϕ, $val(A, \phi) = val(A', \phi)$ holds. Finally, it is easily verified that for any meta theory Σ and meta proposition ψ,

$$
\begin{array}{rll}
\Sigma{\models\!\!\!\approx}\psi & \text{iff} & \forall \phi \in \Phi_\Sigma.\ \phi^A {\models\!\!\!\approx} \psi \\
 & \text{iff} & \forall \phi \in \Phi_\Sigma.\ \phi^A \cup \{\neg\psi\} \text{ unsatisfiable}
\end{array}
\tag{47}
$$

In the following section we will present the method roughly presented in [1] for determining the lower bound of a letter p. We then generalise the method to arbitrary propositions (Section 4.2.2), while in Section 4.2.3 we will present an alternative one.

4.2.1 Determining the greatest lover bound of a literal

For a propositional letter p and meta theory Σ, we first address the problem to compute $glb(\Sigma, p)$ and $glb(\Sigma, \neg p)$.

By Equation (47), we know that for any $n \in [0, 1]$,

$$
\Sigma{\models\!\!\!\approx}(p{\geq}n) \quad \text{iff} \quad \forall \phi \in \Phi_\Sigma.\ \phi^A \cup \{(p{<}n)\} \text{ not satisfiable}
\tag{48}
$$

In order to compute $glb(\Sigma, p)$, we are looking for the largest n such that Equation (48) holds. It is quite easy to see that for any $\phi \in \Phi_\Sigma$, $\phi^A \cup \{(p{<}pos(p, \phi))\}$ is unsatisfiable and there is no $m > n$ such that $\phi^A \cup \{(p{<}m)\}$ is unsatisfiable too, i.e. for any $\phi \in \Phi_\Sigma$ and letter p, $pos(p, \phi)$ is the largest value n that can be chosen making $\phi^A \cup \{(p{<}n)\}$ unsatisfiable. As a consequence, $glb(\Sigma, p) = \min\{val(p, \phi) : \phi \in \Phi_\Sigma\}$ holds. In a quite similar way it can be verified that $glb(\Sigma, \neg p) = \min\{val(\neg p, \phi) : \phi \in \Phi_\Sigma\}$. Therefore, for any literal l,

$$
glb(\Sigma, l) = \min\{val(l, \phi) : \phi \in \Phi_\Sigma\}
\tag{49}
$$

i.e. $glb(\Sigma, l)$ can directly be computed from the completed branches $\phi \in \Phi_\Sigma$.

4.2.2 Determining the greatest lover bound of a proposition

We extend the above idea to the general case. Let us consider a proposition A in CNF, i.e. $A = C_1 \wedge \ldots \wedge C_m$, where each C_i is $l_{i_1} \vee \ldots \vee l_{i_{k_i}}$ and l_{i_j} literal and $l_{i_h} \neq l_{i_l}$ holds for all $h \neq l$, i.e. no literal appears more than once in a clause. By Equation (25), it follows that for any $n \in [0, 1]$,

$$
\begin{array}{rll}
\Sigma{\models\!\!\!\approx}((\bigwedge_i C_i){\geq}n) & \text{iff} & \Sigma{\models\!\!\!\approx} \bigwedge_i (C_i{\geq}n) \\
 & \text{iff} & \text{AND}_i \quad \Sigma{\models\!\!\!\approx}(C_i{\geq}n)
\end{array}
\tag{50}
$$

Therefore,

$$glb(\Sigma, \bigwedge_i C_i) \quad = \quad \min_i\{glb(\Sigma, C_i)\} \tag{51}$$

For any clause C_i and for any $n \in [0,1]$, by Equations (47) and (24)–(26)

$$\Sigma {\mid\!\approx} (C_i \geq n) \quad \text{iff} \quad \forall \phi \in \Phi_\Sigma.\ \phi^A \cup \{(l_{i_1} < n), \ldots, (l_{i_{k_i}} < n)\} \text{ unsatisfiable} \tag{52}$$

By defining $n_i^\phi \in [0,1]$ as the maximal value n for which $\phi^A \cup \{(l_{i_1} < n), \ldots, (l_{i_{k_i}} < n)\}$ is unsatisfiable, i.e. $n_i^\phi = \max\{n : \phi^A \cup \{(l_{i_1} < n), \ldots, (l_{i_{k_i}} < n)\} \text{ unsatisfiable}\}$, it follows that

$$glb(\Sigma, C_i) \quad = \quad \min\{n_i^\phi : \phi \in \Phi_\Sigma\} \tag{53}$$

So, let us find n_i^ϕ. We have to distinguish two cases: (i) for no letter p, both $p, \neg p \in \{l_{i_1}, \ldots, l_{i_{k_i}}\}$; and (ii) otherwise, there is some p such that both $p, \neg p \in \{l_{i_1}, \ldots, l_{i_{k_i}}\}$. Both cases are an extension of the simple case discussed in the previous section. With respect to case (i), it is easily verified that the maximal value n_i^ϕ for which $\phi^A \cup \{(l_{i_1} < n_i), \ldots, (l_{i_{k_i}} < n_i)\}$ is unsatisfiable is determined by

$$\begin{aligned} n_i^\phi &= \max\{n : \phi^A \cup \{(l_{i_1} < n), \ldots, (l_{i_{k_i}} < n)\} \text{ unsatisfiable}\} \\ &= \max\{val(l_{i_j}, \phi) : l_{i_j} \in \{l_{i_1}, \ldots, l_{i_{k_i}}\}\} \\ &= val(C_i, \phi) \end{aligned} \tag{54}$$

Case (ii) is complicated by the fact that if for some p, both $p, \neg p \in \{l_{i_1}, \ldots, l_{i_{k_i}}\}$ then we have to take into account that

$$\begin{aligned} \max\{n : \{(p < n), (\neg p < n)\} \text{ not satisfiable}\} &= \\ \max\{n : \{(p < n), (p > 1-n)\} \text{ not satisfiable}\} &= 0.5 \end{aligned} \tag{55}$$

As a consequence, in case (ii)

$$\begin{aligned} n_i^\phi &= \max\{n : \phi^A \cup \{(l_{i_1} < n), \ldots, (l_{i_{k_i}} < n)\} \text{ unsatisfiable}\} \\ &= \max\{0.5, val(l_{i_j}, \phi) : l_{i_j} \in \{l_{i_1}, \ldots, l_{i_{k_i}}\}\} \\ &= \max\{0.5, val(C_i, \phi)\} \end{aligned} \tag{56}$$

We have, thus, a simple method to compute $glb(\Sigma, A)$, where $A = \bigwedge_{i=1}^m C_i$:

1. determine the set of completed branches of Σ, Φ_Σ;
2. if $\Phi_\Sigma = \emptyset$, then $glb(\Sigma, A) = 1$;
3. otherwise, for each $\phi \in \Phi_\Sigma$ and for each C_i, compute n_i^ϕ according to Equations (54) and (56). Let $n^\phi = \min\{n_1^\phi, \ldots, n_m^\phi\}$;
4. then,

$$glb(\Sigma, A) \quad = \quad \min\{n^\phi : \phi \in \Phi_\Sigma\} \tag{57}$$

It is worth noting that due to Equation (56),

$$n^\phi \quad \geq \quad val(A, \phi) \tag{58}$$

holds and, thus,

$$glb(\Sigma, A) \quad \geq \quad \min\{val(A, \phi) : \phi \in \Phi_\Sigma\} \tag{59}$$

The following example illustrates how the above procedure works.

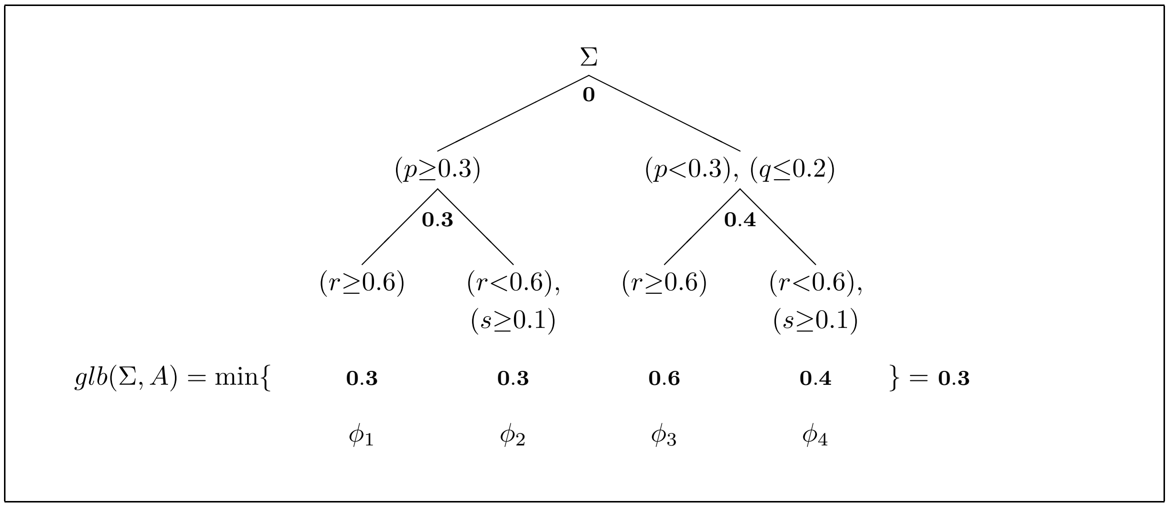
Example 5 Consider the following meta theory

$$\Sigma = \{(p{\geq}0.3) \vee (q{\leq}0.2), (r{\geq}0.6) \vee (s{\leq}0.1), (r{\geq}0.4)\}$$

and the proposition

$$A = (p \vee \neg q) \wedge (p \vee r) \wedge (q \vee \neg q) \wedge (q \vee r)$$

Figure 2 shows a deduction tree produced in order to compute Φ_Σ.



Σ
0
$(p{\geq}0.3)$
$(p{<}0.3)$, $(q{\leq}0.2)$
0.3
0.4
$(r{\geq}0.6)$
$(r{<}0.6)$, $(s{\geq}0.1)$
$(r{\geq}0.6)$
$(r{<}0.6)$, $(s{\geq}0.1)$
$glb(\Sigma, A) = \min\{$ **0.3** **0.3** **0.6** **0.4** $\} =$ **0.3**
ϕ_1 ϕ_2 ϕ_3 ϕ_4

Figure 2: Deduction tree for $glb(\Sigma, A)$ and A in CNF.

Additionally, at each node, we report the value of n^ϕ in bold. It can easily be verified that $glb(\Sigma, A) = 0.3$ ■

In Table 4 the detailed algorithm for computing $glb(\Sigma, A)$ is presented, which has been generalised to the case where the proposition A is in NNF. It is based on SAT. With respect to the SAT algorithm, there are some items which have further to be clarified, namely Step 1, 3, 4d and 4e.

Step 1 expresses the termination condition of the algorithm: either $glb = 0$ and, thus, for no $n > 0$, $\Sigma \mathrel{|\!\not\approx} (A{\geq}n)$ holds, or $\Phi = \emptyset$, i.e. there are no branches which can be further be expanded. In this latter case, $glb = 2$ means that each branch ϕ is closed, and, thus Σ is not satisfiable and consequently $glb(\Sigma, A) = 1$.

Step 3 implements a optimisation heuristics: indeed, at each step of the algorithm, we evaluate the truth value of A with respect of the current branch ϕ. If this value is already greater than the current glb, then we can stop the expansion of ϕ. In fact, we already know that for each complete expansion ϕ' of ϕ, at Point 4e, we will have that $\min\{glb, val(A, \phi')\} = glb$[4].

Step 4d is more tricky and has to do with the fact that A is not necessarily in CNF. If A is in CNF then Step 4d is not required, but at Step 3a, the evaluation of A is determined by n^ϕ according Equations (54) and (56). If A is not in CNF then we have to consider the case (ii) above (see Equation (56)) where a conjunct of a CNF of A contains both p and $\neg p$, for a letter p. To this end we add (the axiom) $(p{\geq}0.5) \vee (p{<}0.5)$ to the branch. The conditions $(4(d)i)$–$(4(d)iv)$ just limit the application of this rule: conditions $(4(d)i)$ and $(4(d)iv)$ are self explanatory; concerning condition $(4(d)ii)$, if $glb(\phi) \geq 0.5$ then by adding $(p{\geq}0.5) \vee (p{<}0.5)$ to ϕ^M, still $glb(\phi) = val(A, \phi) = val(A, \phi') = glb(\phi')$ holds; and condition $(4(d)ii)$ should hold otherwise $(p{\geq}0.5) \vee (p{<}0.5)$ is already entailed by ϕ^M, i.e. $\phi^M \mathrel{|\!\approx} (p{\geq}0.5) \vee (p{<}0.5)$.

Finally, Point 4e manages the case a completed branch ϕ is reached. In this case the truth value of A with respect to ϕ is evaluated and the current glb is updated.

[4]Something similar has been used in [8].

Algorithm minmaxglb(Σ, A)**:**

minmaxglb(Σ, A) starts from the root labelled Σ. Initialise Φ with $\Phi = \{\phi\}$, where $\phi^M = \Sigma$. Φ is managed as a multiset. Let $glb \leftarrow 2$. The variable glb will contain the value of $glb(\Sigma, A)$.

```
1. if $\Phi = \emptyset$ or $glb = 0$ then
    (a) if $\Phi = \emptyset$ and $glb = 2$ then let $glb \leftarrow 1$;
        /* $\Sigma$ not satisfiable */
    (b) return $glb$ and exit;
2. otherwise, select a branch $\phi \in \Phi$ and remove it from $\Phi$, i.e. $\Phi \leftarrow \Phi \setminus \{\phi\}$;
3. if $glb \neq 2$ then
    (a) evaluate $A$ with respect to $\phi$, i.e. let $glb(\phi) \leftarrow val(A, \phi)$;
    (b) if $glb(\phi) \geq glb$ then go to step 1. ($\phi$ needs not to be expanded anymore);
4. try to apply a rule to $\phi$ with the following priority among the rules: $(up) \succ (\wedge) \succ (\vee)$:
    (a) if the $(up)$ rule is applicable to $\phi$ then expand $\phi$ by repeated application of the $(up)$
        rule until $(up)$ is no more applicable. Let $\phi'$ be the resulting branch. If $\phi'$ is not closed
        then add it to $\Phi$, i.e. $\Phi \leftarrow \Phi \cup \{\phi'\}$. Go to step 1.
    (b) if the $(\wedge)$ rule is applicable to $\phi$ then expand $\phi$ by repeated application of the $(\wedge)$ rule
        until $(\wedge)$ is no more applicable. Let $\phi'$ be the resulting branch. If $\phi'$ is not closed then
        add it to $\Phi$, i.e. $\Phi \leftarrow \Phi \cup \{\phi'\}$; Go to step 1.
    (c) if the $(\vee)$ rule is applicable to $\phi$ then expand $\phi$ by *one* application of the $(\vee)$ rule. Let
        $\phi_1$ and $\phi_2$ be the resulting branches. For each $\phi_i, i = 1, 2$, if $\phi_i$ is not closed then add
        it to $\Phi$, i.e. $\Phi \leftarrow \Phi \cup \{\phi_i\}$. Go to step 1.
    (d) if $p$ is a letter such that the following conditions hold:
        i. $p$ and $\neg p$ occur in $A$;
        ii. if $glb \neq 2$ then $glb(\phi) < 0.5$;
        iii. $val(p, \phi) < 0.5$ and $val(\neg p, \phi) < 0.5$;
        iv. $(p \geq 0.5) \vee (p < 0.5) \notin \phi^M$;
        then extend branch $\phi$ by adding $(p \geq 0.5) \vee (p < 0.5)$ to $\phi^M$. Let $\phi'$ be the resulting
        branch, i.e. $\phi'^M = \phi^M \cup \{(p \geq 0.5) \vee (p < 0.5)\}$ and add $\phi'$ to $\Phi$, i.e. $\Phi \leftarrow \Phi \cup \{\phi'\}$. Go
        to step 1.
    (e) otherwise, update $glb$, i.e. let $glb \leftarrow \min\{glb, val(A, \phi)\}$
        /* $\phi$ is completed */
```

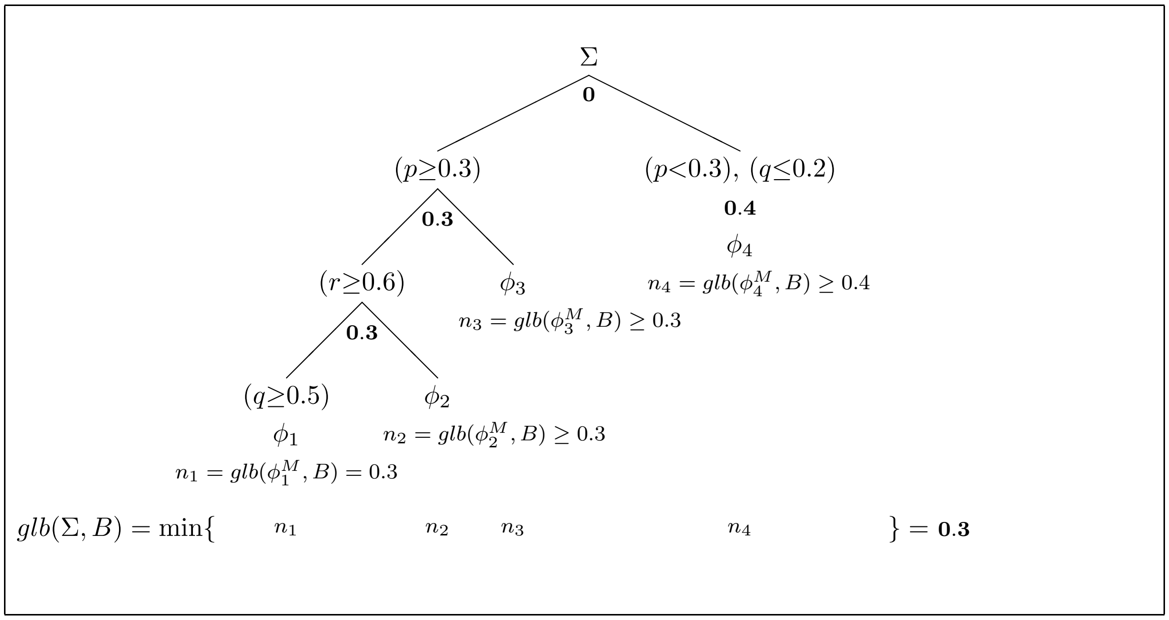
Table 4: The minmaxglb procedure.

The following example shows how minmaxglb works.

Example 6 Consider the meta theory Σ and the proposition A of Example 5. Let B be the proposition

$$B \quad = \quad (p \wedge q) \vee (\neg q \wedge r)$$

B is an equivalent DNF of proposition A. Figure 3 shows a deduction tree generated executing $mimmaxglb(\Sigma, B)$. Additionally, at each node, we report the value of $glb(\phi)$ in bold. It is worth



Σ
0
$(p{\geq}0.3)$
$(p{<}0.3)$, $(q{\leq}0.2)$
0.3
0.4
ϕ_4
$(r{\geq}0.6)$
ϕ_3
$n_4 = glb(\phi_4^M, B) \geq 0.4$
0.3
$n_3 = glb(\phi_3^M, B) \geq 0.3$
$(q{\geq}0.5)$
ϕ_2
ϕ_1
$n_2 = glb(\phi_2^M, B) \geq 0.3$
$n_1 = glb(\phi_1^M, B) = 0.3$
$glb(\Sigma, B) = \min\{ \; n_1 \; n_2 \; n_3 \; n_4 \; \} =$ **0.3**

Figure 3: Deduction tree for $mimmaxglb(\Sigma, B)$.

noting that branch ϕ_2, ϕ_3 and ϕ_4 need not to expanded further, according to Step 3.

■

4.2.3 Determining the greatest lover bound of a proposition: an alternative approach

In this section we present an alternative method to compute $glb(\Sigma, A)$. The method is based on the fact that from Σ it is possible to determine a finite set $N^{\Sigma} \subset [0, 1]$, where $|N^{\Sigma}|$ is $O(|\Sigma|)$, such that $glb(\Sigma, A) \in N^{\Sigma}$, i.e. $glb(\Sigma, A)$ has to be an element of N^{Σ}. Therefore, $glb(\Sigma, A)$ can be determined by computing the greatest value $n \in N^{\Sigma}$ such that $\Sigma \cup \{(A{<}n)\}$ is unsatisfiable, i.e. $\mathsf{SAT}(\Sigma \cup \{(A{<}n)\})$ returns `false`. The easiest way to do that is to order the elements of N^{Σ} and the to perform a binary search among these values.

Let us determine N^{Σ}. As $\Sigma {\approx\!\!\!\!|}\, (A{\geq}0)$, $0 \in N^{\Sigma}$ follows. Additionally, Σ could be not satisfiable and, thus, $\Sigma {\approx\!\!\!\!|}\, (A{\geq}1)$, i.e. $1 \in N^{\Sigma}$. Since $glb(\Sigma, A) = glb(\Sigma, A')$ for any equivalent CNF A' of A, without loss of generality we will assume that A is in CNF, i.e. $A = \bigwedge_{i=1}^{m} C_i$. By Equation (57), $glb(\Sigma, A)$ is one of the values n^{ϕ} such that $\phi \in \Phi_{\Sigma}$, where $n^{\phi} = \min\{n_1^{\phi}, \ldots, n_m^{\phi}\}$ and each n_i^{ϕ} is determined according to Equations (54) and (56). Therefore, $glb(\Sigma, A)$ is 0.5 (thus, $0.5 \in N^{\Sigma}$) or one of the values $val(C_i, \phi)$. But, $val(C_i, \phi)$ is one of the values $val(l_{i_j}, \phi)$ which is determined by $pos(p, \phi)$ or $neg(p, \phi)$, according whether l_{i_j} is p or $\neg p$, respectively (see Equation (46)). Finally, we have to determine the possible values of $pos(p, \phi)$ and $neg(p, \phi)$, respectively. That is, by Equation (45), we have to determine the value n for which $(p{\geq}n) \in \phi^A$ or $(p{>}n) \in \phi^A$ and the value $1 - m$ for which $(p{\leq}m) \in \phi^A$ or $(p{<}m) \in \phi^A$. Since each $\psi \in \Sigma$ is in NNF and given the inference rules used by SAT (see Table 2), it follows that each number n occurring in Σ and its

```
Algorithm binaryglb(Σ, A):
Let Σ be a meta theory and let A be a proposition. Set $Min := 0$ and $Max := 2$.
  1. determine $N^{\Sigma}$ according to Equation (60);
  2. pick binary $n \in N^{\Sigma}$ such that $Min < n < Max$. If there is no such $n$, then set $glb(\Sigma, A) := Min$ and exit;
  3. if $\Sigma \approx\!\!\mid (A \geq n)$ then set $Min = n$, else set $Max = n$. Go to step 2.   ■
```

Table 5: The binaryglb procedure.

negation $1 - n$ are possible values for $glb(\Sigma, A)$. For instance, for the meta theory in Example 6, the possible values for $glb(\Sigma, A)$ are $\{0, 0.5, 1\} \cup \{0.3, 0.7, 0.2, 0.8, 0.6, 0.4, 0.1, 0.9\}$. The reason to consider, e.g., $0.7 = 1 - 0.3$ is given by the use of the $(\vee)$ rule applied to $(p \geq 0.3) \vee (q \leq 0.2)$ which generates a branch containing $(p < 0.3)$. This observation leads us directly to a way to be even more precise. In fact, consider the set of rules $\mathcal{R}^s = \{(up), (\wedge), (\vee_s)\}$. Obviously, SAT is still correct and complete using $\mathcal{R}^s$ in place of $\mathcal{R}^{DPLL}$. But then, if e.g. $(p < m) \in \phi^A$ then $(p < m)$ has to occur in Σ as meta literal. Similarly for the other cases. So, for a meta theory Σ, we define

$$\begin{array}{rcl} N^{\Sigma} & = & \{0, 0.5, 1\} \cup \\ & & \{n : (p \geq n) \text{ or } (p > n) \text{ occurs in } \Sigma\} \cup \\ & & \{1 - n : (p \leq n) \text{ or } (p < n) \text{ occurs in } \Sigma\} \end{array} \tag{60}$$

For instance, for the meta theory in Example 6, N^{Σ} is given by $\{0, 0.5, 1\} \cup \{0.3, 0.8, 0.6, 0.9, 0.4\}$. The following proposition follows from the discussion above.

Proposition 5 *Let Σ be a meta theory. Then $glb(\Sigma, A) \in N^{\Sigma}$.* ⊣

Note that, since for any proposition A, $(A < 0.5)$ is normalised, it follows from Corollary 2 that $\models A$ iff $\approx\!\!\mid (A \geq 0.5)$, i.e.the truth value of classical propositional tautologies is at least 0.5. But, by Proposition 5, $glb(\emptyset, A) \in \{0, 0.5\}$ and, thus, $\models A$ iff $glb(\emptyset, A) = 0.5$, i.e. classical tautologies have 0.5 as its greatest truth value lover bound.

As anticipated, Table 5 specifies the algorithm for determining $glb(\Sigma, A)$ through a binary search among the values in N^{Σ}. The value of $glb(\Sigma, A)$ can, thus, be determined in at most $\log |N^{\Sigma}|$ entailment tests, i.e. SAT calls.

Example 7 Consider Example 6. It follows that $N^{\Sigma} = \{0, 0.3, 0.4, 0.5, 0.6, 0.8, 0.9, 1\}$. The following table highlights a binaryglb(Σ, B) execution.

Round	n	Min	Max	$\Sigma \approx\!\!\mid (B \geq n)$
0	–	0	2	–
1	0.5	0	2	`false`
2	0.3	0	0.5	`true`
3	0.4	0.3	0.5	`false`
4	–	0.3	0.4	–

After three entailment tests the procedure stops as there is no value $n \in N^{\Sigma}$ within Min and Max and, thus, the result is returned: $glb(\Sigma, A) := Min = 0.3$.

■

We finish this section by addressing the computational complexity of determining $glb(\Sigma, A) \geq n$.

Proposition 6 *Let Σ be a meta theory, let A be proposition and $n \in [0, 1]$. Then checking whether $glb(\Sigma, A) \geq n$ is a coNP-complete problem.* ⊣

Proof: Let $n \in [0,1]$. Since $\Sigma \not\approx (A \geq n)$ iff $glb(\Sigma, A) \geq n$, and (see Corollary 3) for all n, deciding $\Sigma \not\approx (A \geq n)$ is a coNP-complete problem, coNP-hardness of the $glb(\Sigma, A) \geq n$ decision problem follows.

The following NP algorithm determines whether $glb(\Sigma, A) < n$. Non-deterministically generate a deduction tree T for $\Sigma' = \Sigma \cup \{(A{<}n)\}$ by running $\mathsf{SAT}(\Sigma')$: $glb(\Sigma, A) < n$ iff there is some completed branch ϕ in T. The depth of the branch ϕ is polynomially bounded by the input. Hence, deciding $glb(\Sigma, A) < n$ is in NP. Therefore, deciding $glb(\Sigma, A) \geq n$ is in coNP. `Q.E.D.`

5 Evaluation

In the following two sections we will evaluate the performance of the presented algorithms. In particular, in Section 5.1 we will address the fuzzy SAT problem, while in Section 5.2 we will deal with the BTVB problem.

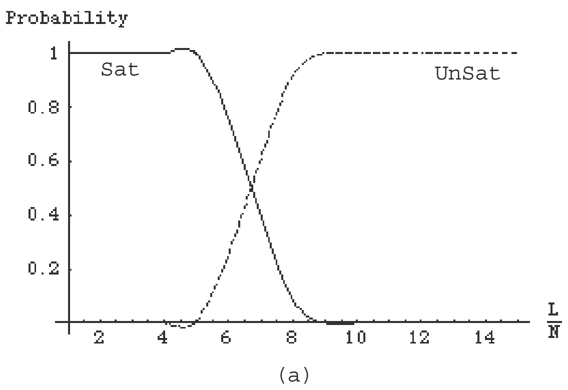
5.1 The fuzzy SAT problem

The evaluation method that we adapt is an extension to $\mathcal{L}^M$ of the one usually used in classical SAT experiments and proposed by Mitchell *et al.* [11]. To set up a benchmark suite for DPLL theorem provers Mitchell *et al.* generate propositional formulae using the *fixed clause-length* model. Each random proposition is a proposition in CNF. There are several parameters which guide the generation of CNFs: (*i*) the number N of propositional letters; (*ii*) the number K of literals in a disjunct; and (*iii*) the number L of conjuncts. Based on a given choice of the above parameters, *random K-CNF* propositions are generated as follows. A *random letter* is a variable randomly chosen from the set of propositional letters. A *random literal* is with probability 0.5 a random letter or its negation, otherwise. A *random K-CNF clause* is a disjunction of K random literal such that no letter appears more than once. The extension to the meta level is as follows. A *random K-CNF meta clause* is of the form $(A{\geq}n)$, where A is a random K-CNF clause and n is a randomly chosen value in $(0,1)$. Each random K-CNF meta clause expresses a constrain on the truth value of its literal. A *random K-CNF meta proposition* is a conjunction of L random K-CNF meta clauses. Finally, a *random K-CNF meta theory* is a set of L random K-CNF meta clauses. As the application of the $(\wedge)$ rule to a random K-CNF meta proposition yields a random K-CNF meta theory, we will restrict the test to random K-CNF meta theories only. All parameters except L, the number of clauses, are fixed (e.g., $N = 30$ and $K = 3$). The parameter L ranges from N to $15N$. For each value of the ratio L/N a set of 100 random K-CNF meta theories is generated. For each generated meta theory Σ we measure both the time t needed to determine satisfiability and the number of nodes s (in the following called *states*) of the deduction tree. Since checking a single meta theory can take arbitrarily long in the worst case, there is an upper limit for the CPU time consumed. As soon as the upper limit is reached, the computation for Σ is stopped. We set this upper limit to 2000 seconds. For each ratio $r = L/N$, $r = 1, \ldots, 15$, both the median time t_r of the 100 tests and the median number of states s_r are determined[5]. The *time performance graph* (*state performance graph*) has as X-axis the values of r and as Y-axis the values of t_r (of s_r). In case of evaluation comparison, the seed of the random generator has been maintained identical so that identical problem instances are generated. Our tests have been run on a SUN Ultra 5 with 384 MB main memory and, of course, a depth first strategy has been adopted.

Figure 4 reports the result of our tests, where $N = 10, 20, 30, K = 3$. In it, (a) reports the probability of satisfiability and unsatisfiability of meta theories for $N = 30, K = 3$, (b) reports the median CPU time and (c) reports the median number of states. In (b) we also report the performance of SAT, where the (*up*) rule has been replaced with the simpler version $(\perp)$. The

[5]Given $T_r = t_{r_1}, \ldots, t_{r_{100}}$, let $T_r^{\leq} = t'_{r_1}, \ldots, t'_{r_{100}}$ be T ordered according to $\leq$. Then t_r is $(t'_{50} + t'_{51})/2$. Note that the mean of a set of number is influenced by a very small number of very large values. As the median is less sensitive to such "out-liers", it appears to be a more informative statistics.

graphs show similar patterns to the well known patterns for classical propositional logic [11, 15]. In the following two paragraphs we will comment the results.



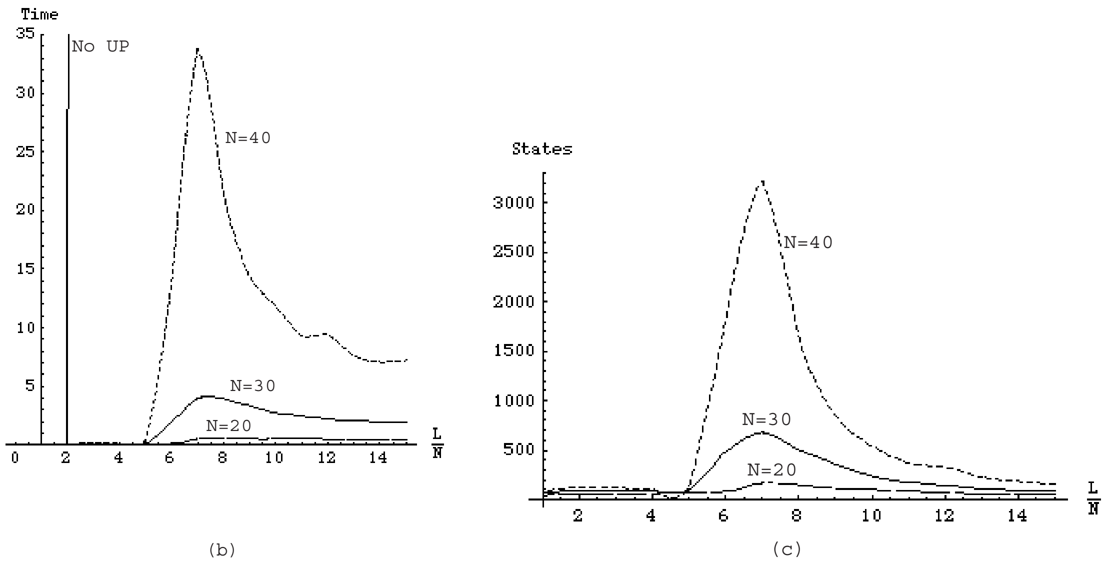


Figure 4: (a) Probability of theory satisfiability vs. theory unsatisfiability, for $K = 3, N = 30$. (b-c) Satisfiability tests for random K-CNF meta theories, for $K = 3, N = 20, 30, 40$.

Satisfiability vs. unsatisfiability: Figure 4a tells us that small random 3-CNF meta theories ($r \leq 6$) are likely to be satisfiable, whereas large random 3-CNF meta theories ($r \geq 8$) are not, i.e., the probability that a random 3-CNF meta theory is satisfiable decreases as the number of clauses increases. This behaviour is motivated by the fact that the more clauses are in a theory Σ the more constrains on the truth value of the literal appearing in Σ there are and, thus, the less completed branches $\phi \in \Phi_\Sigma$ (from which models of Σ can be build[6]) there are. The unsatisfiability graph is complementary to the satisfiability one. The crossing point between the two graphs is the point where the probability of satisfiability is 0.5 and is called *phase transition point* (*ptp*). As described below, this is the point where the *hard satisfiability problems* are. Through interpolation, we estimated the point at $ptp \approx 6.70$ which is greater than the value for classical propositional logic ($ptp =\approx 4.3$). This is motivated by the fact that there are satisfiable meta propositions ψ for which ${}^c\psi$ is not satisfiable. For instance, as we already pointed out, $(A{\geq}0.3) \wedge (A{\leq}0.7)$ is

[6]Note that a meta theory Σ may have an infinite number of models.

satisfiable whereas $A \wedge \neg A$ is classically unsatisfiable. Therefore, in order to make a meta theory Σ unsatisfiable a higher number of clauses is required with respect to the classical propositional case. In Figure 5 we report the same test except that normalised random 3-CNF meta theories have been generated. As we can see *ptp* is located at ≈ 4.48 which is compliant both to the classical case and to Proposition 3.

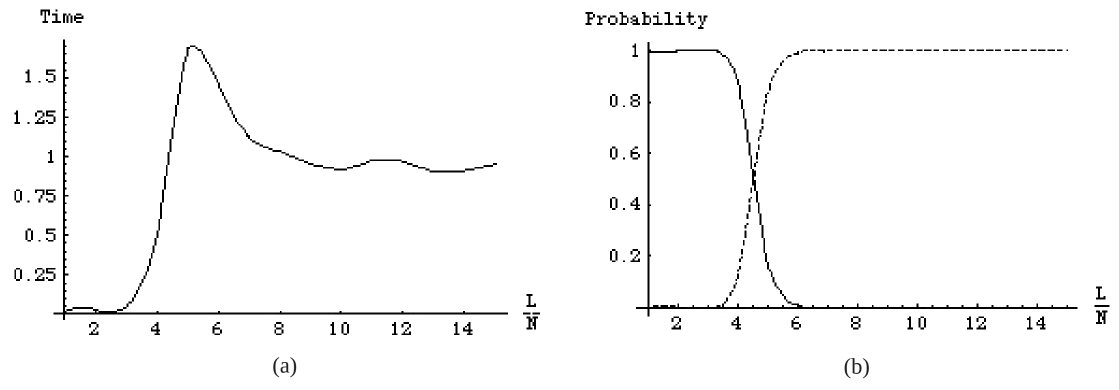


Figure 5: Satisfiability test for normalised random 3-CNF meta theories and $N = 30$.

Phase transition: Figure 4b reports the median CPU time required for each ratio $r = L/N$, $r = 1, \ldots, 15$, in case of using either $(\perp)$ or (up). What (b) shows is that in order to perform any successful experiment the (up) rule should be used. For classical propositional logic this fact is well known and, $\mathcal{L}^M$ is no exception.

More interestingly, (b) shows the typical *easy-hard-easy* pattern. For random 3-CNF meta theories that are either small or relatively large SAT finishes quickly, but random 3-CNF meta theories of medium size take much longer. Since random 3-CNF meta theories with few clauses are *under constrained* and have many completed branches, a completed branch is likely to be found early in the search. Random 3-CNF meta theories with very many clauses are *over constrained* (and usually unsatisfiable), so contradictions are found easily, so a full search can be completed quickly. Finally, random 3-CNF meta theories in between are much harder because they have relatively few (if any) completed branches (see Figure 6), and a closed branch ϕ will only be generated after assigning truth value constraints ($pos(p, \phi)$ and $neg(p, \phi)$) to a large number of letters, resulting in a deep search tree. As for classical SAT problems, the hard area is related to the probability of satisfiability. Indeed, the peak of the median CPU time, as well as for the median number of states, closely corresponds to the point where the probability of satisfiability is 0.5 (see also [16]).

5.2 The BTVB problem

In this section we will evaluate the two algorithms presented, minmaxglb and binaryglb, to compute the greatest lower bound $glb(\Sigma, A)$ for a meta theory Σ and a proposition A. We will extend the test method for the fuzzy SAT problem to the BTVB problem as follows. Each meta theory Σ will be a random K-CNF meta theory, whereas each proposition A, for which we will compute its greatest lower bound, will be a proposition in DNF is defined as follows: (i) a *random K-DNF clause* is a conjunction of K random literal such that no letter appears more than once; and (ii) a *random K-DNF proposition* is a disjunction of random K-DNF clauses. As for the satisfiability problem, the values for N and K are fixed, whereas the parameter L ranges from N to $15N$. For each value of the ratio $r = L/N$, $r = 1, \ldots, 15$, and for each $i = 1, \ldots, 100$ we generate a random K-CNF meta theories Σ_i^r of L clauses and a random K-DNF proposition A_i^r (of $N \leq l_i^r \leq 15N$ clauses) for which $glb(\Sigma_i^r, A_i^r)$ should be determined. For each pair (r, i) we measure the time t_i^r

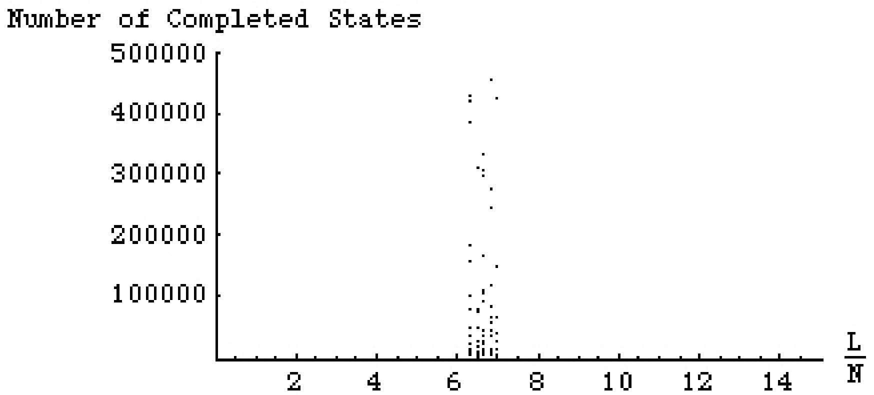


Figure 6: Number of completions for $r = L/N \in [6,7]$ and $K = 3, N = 30$.

and the median number of states s_i^r required and compute for each ratio r the median time t_r and number of states s_r as for the fuzzy SAT problem. The following two points have to be clarified about the choice of A_i^r:

1. A_i^r is in DNF. This is due to the fact that in this way it is easier to randomly generate a proposition A for which $glb(\Sigma, A) > 0$, i.e. there is an $n > 0$ such that $\Sigma \approx\!\!\!\mid (A \geq n)$. Indeed, we tried several algorithms for generating randomly propositions in CNF, but in almost all cases $glb(\Sigma, A)$ was 0. Furthermore, a benefit of considering random K-DNF propositions A is that in case of $\mathsf{binaryglb}$, by invocation of $\mathsf{SAT}(\Sigma \cup \{(A<n)\})$, with value $n \in N^{\Sigma}$, $(A<n)$ is, after NNF transformation (see, (25),(26)), already in CNF;

2. the number l_i^r of random K-DNF clauses in A_i^r is not fixed within a sample of 100 tests. Indeed, l_i^r is uniformly distributed in $[N, 15N]$, the range of theory length. So, for each r and $i = 1, \ldots, 100$, $l_i^r = \lfloor \frac{i(15N-1)}{100} \rfloor + 1$.

Further, we slightly modify $\mathsf{binaryglb}$ as follows: (i) we remove 0 and 1 from N^{Σ}; (ii) then we check first whether $\Sigma \approx\!\!\!\mid (A \geq m)$, where $m = \min\{n : n \in N^{\Sigma}\}$ holds. If not then we know that $glb(\Sigma, A) = 0$, so we can stop the search. Otherwise, we test if Σ is satisfiable. If not then $glb(\Sigma, A) = 1$, else we proceed with a binary search among the remaining values in N^{Σ}. The idea is to reduce the number of SAT calls in all those cases where either the theory is unsatisfiable or the greatest lower bound is 0.

In Figures 7-9 the results of our tests are shown.

In Figure 7a, for $K = 3, N = 30$ we report the statistics of the computed greatest lower bounds. As the random K-CNF meta theories are as for the satisfiability test, the `Sat` and `UnSat` curves behave exactly identical ($ptp \approx 6.7$). Concerning the glb values, four statistics are reported for each sample $r = 1, \ldots, 15$: (i) the probability $p_1(r)$ of $glb(\Sigma, A) = 1$; (ii) the probability p_0 of $glb(\Sigma, A) = 0$; (iii) the probability $p_{0.5}(r)$ of $glb(\Sigma, A) = 0.5$; and (iv) the probability $p(r)$ of $glb(\Sigma, A) = n \in (0, 0.5) \cup (0.5, 1)$. Of course, $p_1(r) + p_0(r) + p_{0.5}(r) + p(r) = 1$. It is worth noting that, by definition, neither $\top$ nor $\bot$ appear in A and Σ, and both 0 and 1 do not appear in Σ. Further, we restricted random K-CNF meta clause $(B \geq n)$ to the case where $n \neq 0.5$.

(i) Concerning $p_1(r)$, we have that for a random K-DNF proposition A and a random K-CNF meta theory Σ, $glb(\Sigma, A) = 1$ iff Σ is unsatisfiable. So, the `UnSat` curve corresponds also to the case where $glb(\Sigma, A) = 1$.

(ii) Concerning $p_0(r)$, $p_0(0) \approx 0.23$ and decreases as r increases. This is simply motivated by the fact that as the theories grow, the theories become more informative and, thus, the probability that $\Sigma \approx\!\!\!\mid (A \geq n)$ holds, for $n > 0$, grows.

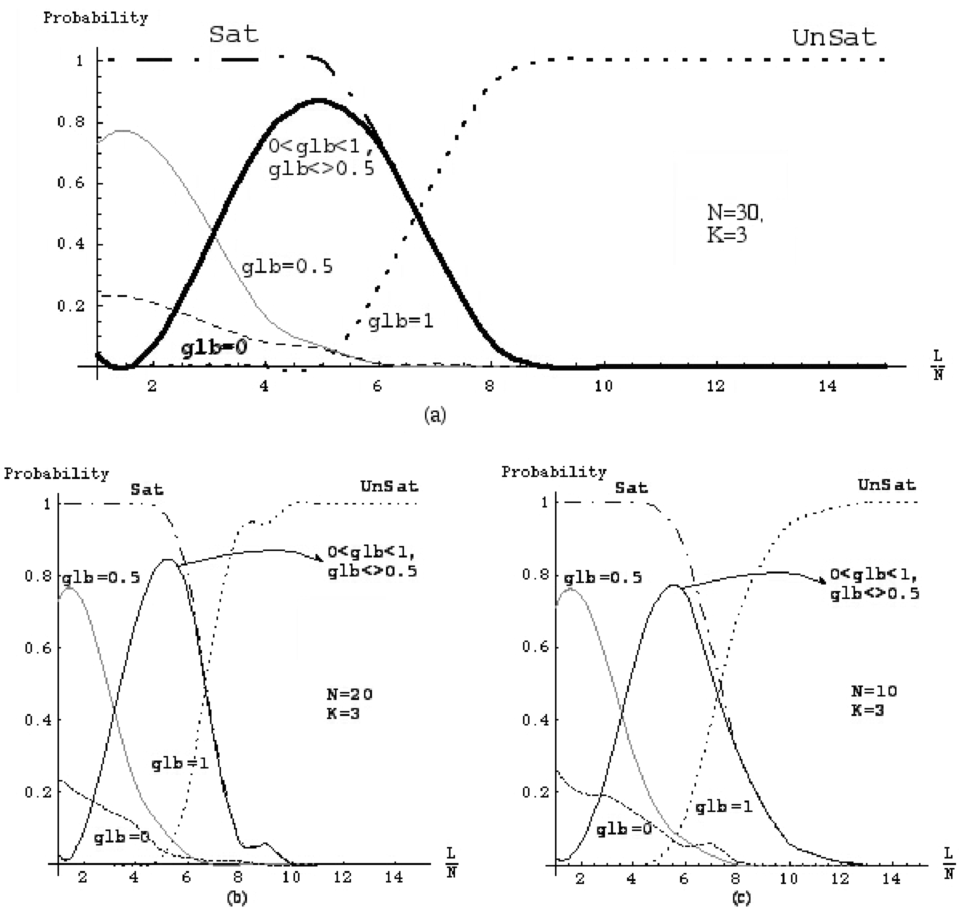
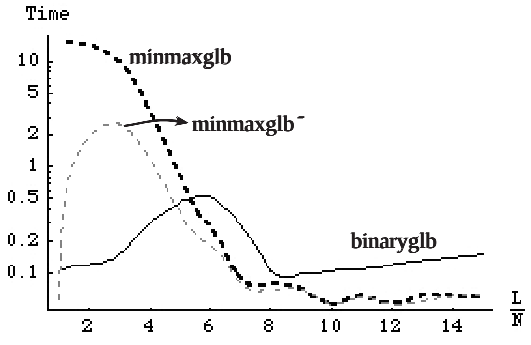


Figure 7: (a-c) Analysis of the computed *glb* values, for $K = 3, N = 30, 20, 10$.

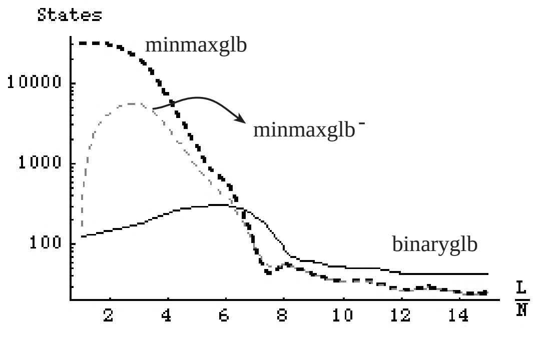
(iii) Concerning $p_{0.5}(r)$, as we have seen early, for a random K-DNF proposition A, A is a classical tautology iff $glb(\emptyset, A) = 0.5$. Since 0.5 does not appear in Σ, $p_{0.5}(0) \approx 0.74$ expresses the probability of having a classical tautology as query among the 100 generated ones. Please note that even if A is a classical tautology this does not mean that $glb(\Sigma, A) = 0.5$, for all Σ. Indeed, $glb(\emptyset, p \vee \neg p) = 0.5$, but $glb(\{(p{\geq}0.8)\}, p \vee \neg p) = 0.8$. Similarly to $p_0(r)$, $p_{0.5}(r)$ decreases as the theories become more informative.

(iv) Concerning $p(r)$, probably the most interesting case, it considers all those cases in which the theory Σ is satisfiable, there is entailment, i.e. $\Sigma|\!\!\approx(A{\geq}n)$ (for $n > 0$) and all those cases in which A is a classical tautology for which Σ contains enough information such that $glb(\Sigma, A) > 0.5$. As the picture shows, $p(r)$'s pattern is that of a normal (or gaussian) distribution with maximum at $r_m \approx 4.9$ and $p(4.9) \approx 0.87$. This is the point where the probability of having entailment is maximal (excluding, the obvious cases of being the theory unsatisfiable). Quite interesting to note is the fact that near to r_m, $p_0(r)$, $p_1(r)$ and $p_{0.5}(r)$ intersect at $r \approx 5.2$ (for which we have no explanation yet). Furthermore, as $p_0(r)$ and $p_{0.5}(r)$ approach to 0, after r_m, $p(r)$ becomes $1 - p_1(r)$ and, thus, behaves as Sat.

In Figures 8, we report the comparison between binaryglb and minmaxglb, for $K = 3, N = 10$. As we can see, binaryglb clearly outperforms minmaxglb in the satisfiability zone. This is due to the fact that in this case each theory has a large set of completions which are computed by minmaxglb. We also reported the curve of the minmaxglb$^-$ algorithm which is as minmaxglb except that Point *4d* has been disabled, as it introduces certainly a lot branches. minmaxglb$^-$ is still correct in the sense that it computes a lower bound, but it is not complete as it does not compute the greatest one. Disabling Point *4d*, reduces the execution time but the comparison to binaryglb is qualitatively the same. As the theories grow, the number of completions decreases and, thus, the execution time of minmaxglb decreases as well. The crossing point between binaryglb and minmaxglb is located at ≈ 5.3 which is close to the r_m point where $p(r)$ is maximal and $p_0(r)$, $p_1(r)$ and $p_{0.5}(r)$ intersect (≈ 5.5, see Figure 7). The difference between the two algorithm is negligible after this point and is motivated by the fact that binaryglb executes in average two SAT calls (first, checking whether there is entailment and second, check for unsatisfiability). What comes out is that the performance of minmaxglb is strictly related to the computation of completions and, thus is slower than binaryglb in the satisfiability zone, while it is comparable to binaryglb both in the *ptp* zone (hard zone) as well as in the unsatisfiability zone. Due to the huge number of completions, we were unable to complete the test for minmaxglb for $K = 3, N \geq 20$.



(d)



(e)

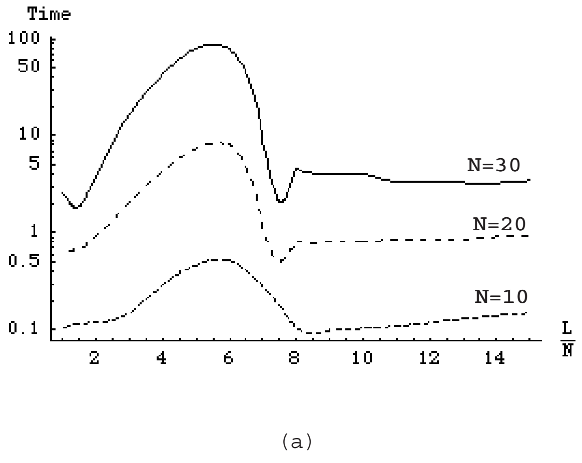
Figure 8: (b,c) Comparison between binaryglb and minmaxglb, for $K = 3, N = 10$.

In Figure 9a, the median time for binaryglb is represented, where, $K = 3$ and $N = 10, 20, 30$. Not surprisingly, (i) the time increases as the number of variable increases; (ii) there is a easy-hard-easy pattern which similar to the satisfiability problem (indeed, binaryglb calls successively SAT and (iii) the hard zone is located near to the point where $p(r)$ is maximal, e.g. for $N = 30$, ≈ 5.7 which is close to $r_m \approx 4.9$. Indeed the hard zone for the greatest lower bound problem is located near to r_m. This is due to the fact that (i) r_m is near to the *ptp* where the satisfiability problem is hardest, so SAT is most expensive; and (ii) since at ≈ 5.7, $p(r)$ is maximal, i.e. the probability that there is entailment is maximal (excluding unsatisfiable theories), this requires in average the maximal number of SAT calls.

Finally, in Figure 9b, the median time for binaryglb is represented, where, $K = 3, N = 30$, but different lengths l_i^r of the query proposition has been considered. Indeed, rather than letting to range the query length from 1 to $15N$, we partitioned $[1, 15N]$ into four parts, $P_1, \ldots, P_4$, where $P_1 = [1, 60], P_2 = [61, 150], P_3 = [151, 240]$ and $P_4 = [241, 450]$. We executed four tests $T_1, \ldots, T_4$. In each test T_j, for each $r = 1, \ldots, 15$, the length l_i^r of the query propositions A is uniformly distributed in P_r. That is, we sampled the queries into *very short, short, medium* and *long* queries in order to compute the impact of the query length to the greatest lower bound problem. As the picture shows, while the easy-hard-easy pattern still remains the same and, in particular, the peeks are still around r_m, the execution time increases notably with increasing query length.

6 Conclusions and future research

In this paper we considered both from a theoretical and practical point of view a well known fuzzy propositional logic, based on Zadeh's implication operator: at the object level, truth values are in the interval $[0, 1]$ and the connectives $\neg, \wedge, \vee$ and $\rightarrow$ are the well known operators $1-$, min, max and Zadeh's implication "$p \rightarrow q = \max\{1 - p, q\}$". At the meta level, restrictions on the truth value of a proposition are allowed by means of the relations $\leq, \geq, <$ and $>$.



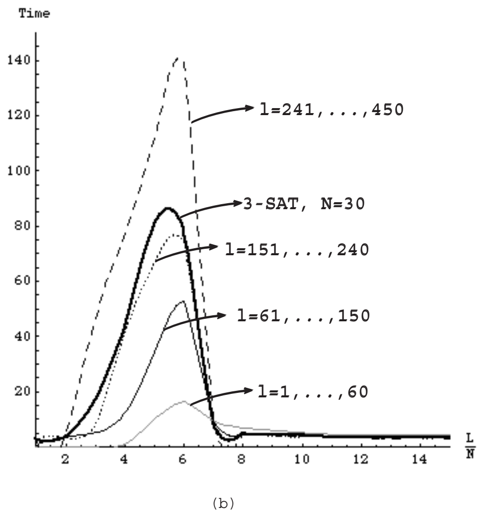


Figure 9: (a) Comparison between binaryglb, for $K = 3, N = 10, 20, 30$. (b) Comparison between binaryglb, for different query lengths, for $K = 3, N = 30$.

We presented several decision algorithms for two important decision problems within it, the fuzzy SAT problem and the BTVB problem, and experimentally tested them.

Concerning the fuzzy SAT problem, essentially it shows a similar easy-hard-easy pattern as for the classical satisfiability problem. Further, as our SAT decision algorithm is a (fuzzy) variant of the classical DPLL SAT algorithm, performance improvements may be obtained by adapting techniques from the classical SAT proof theory (e.g. heuristics for selecting the branching meta literal).

Concerning the BTVB problem, we presented two different algorithms, minmaxglb and binaryglb and have shown that binaryglb clearly outperforms minmaxglb. Interestingly, while binaryglb presents an easy-hard-easy pattern too, the hard zone is located near to the point where the probability of entailment is maximal. From a performance point of view, as binaryglb consists of a reduction of the BTVB problem into several satisfiability problems (logarithmic with respect to the theory size), a main problem for further research consists of reducing this number of calls. Another possibility is to develop an ad-hoc algorithm in the style of [13, 14] which performs the fuzzy entailment test only *once*. Roughly, in order to determine $glb(\Sigma, A)$, we start with $\Sigma' = \Sigma \cup \{(A{<}v)\}$, where v is a new variable symbol. Thereafter, we apply to Σ' deduction rules similar to those in $\mathcal{R}^{DPLL}$ until each branch ϕ_i is completed. Finally, we are looking for the *maximal value* $n \in [0, 1]$ such that for each of the completions ϕ_i, $\phi_i[v/n]$ (if not empty) contains a clash, where $\phi_i[v/n]$ is the set obtained by replacing each occurrence of v by n. Whether this approach may improve the performance has still to be shown and is a topic for further research.

References

[1] Jianhua Chen and Sukhamany Kundu. A sound and complete fuzzy logic system using Zadeh's implication operator. In Zbigniew W. Ras and Michalewicz Maciek, editors, *Proc. of the 9th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-96)*, number 1079 in Lecture Notes In Artificial Intelligence, pages 233–242. Springer-Verlag, 1996.

[2] Marcello D'Agostino and Marco Mondadori. The taming of the cut. Classical refutations with analytical cut. *Journal of Logic and Computation*, 4(3):285–319, 1994.

[3] M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7):394–397, 1962.

[4] M. Davis and H. Putnam. A computing procedure for quantificatio theory. *Journal of the ACM*, 7:201–215, 1960.

[5] Melvin Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, 1990.

[6] Sukhamany Kundu. An imporved method for fuzzy-inferencing using zadeh's implication operator. In *Proceedings of IJCAI Workshops on Fuzzy Logic in AI*, Montreal, Canada, 1995.

[7] Sukhamay Kundu and Jianhua Chen. Fuzzy logic or Lukasiewicz logic: A clarification. In Zbigniew W. Ras and Maria Zemenkova, editors, *Proc. of the 8th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-94)*, number 869 in Lecture Notes In Artificial Intelligence, pages 56–64. Springer-Verlag, 1994.

[8] Jérôme Lang. Semantic evaluation in possibilistic logic. In *Proc. of the 3th Int. Conf. on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-90)*, number 521 in Lecture Notes in Computer Science. Springer-Verlag, 1990.

[9] Richard C. T. Lee. Fuzzy logic and the resolution principle. *Journal of the ACM*, 19(1):109–119, January 1972.

[10] J. Lukasiewicz. *Selected works - Studies in logic and the foundations of mathematics.* Noth-Holland, Amsterdam,Warsaw, 1970.

[11] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distribution of SAT problems. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pages 459–465. AAAI Press/The MIT Press, 1992.

[12] J. Pavelka. On fuzzy logic i,ii,iii. *Zeitschrift für Mathematik und Logik*, 25:45–52,119–134,447–464, 1979.

[13] Umberto Straccia. A four-valued fuzzy propositional logic. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)*, pages 128–133, Nagoya, Japan, 1997.

[14] Umberto Straccia. A fuzzy description logic. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, pages 594–599, 1998.

[15] Special Volume. Frontiers in problem solving: Phase transition and complexity. *Artificial Intelligence Journal*, 81(1–2), 1996.

[16] Toby Walsh. The constrainedness knife-edge. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, pages 406–411, 1988.

[17] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.