On Designing an $\mathcal{L}$-valued Prolog Engine

Raymond Bisdorff
CRP - Centre Universitaire
162a, avenue de la Faïencerie
L-1511 LUXEMBOURG
bisdorff@crpcu.lu

February 16, 2000

Abstract

This text presents our communication at the 9th Benelux Workshop on Logic Programming in Amsterdam, November 20, 1998, where we discussed some theoretical and practical arguments in favour of designing a multi-truth valued extension of Prolog.

Contents

Introduction

This text presents our communication at the 9th Benelux Workshop on Logic Programming in Amsterdam, November 20, 1998.

First we will introduce our $\mathcal{L}$-logic (see [4]), i.e. a symmetric multi-valued logic with split truthfulness and falseness semantics supporting a unique negational fix-point acting as $\mathcal{L}$-undetermined truth value.

Our main contribution is given by a careful distinction we introduce between an underlying propositional credibility calculus and the induced multi-valued truth denotations. This distinction, combined with a special truth/falseness polarization, allows to naturally (in a categorical sense) extend classic bi-valued Boolean algebra to a multi-valued case, i.e. all Boolean tautologies and antilogies remain valid in this $\mathcal{L}$-valued setting.

This result gives some hints toward designing a natural $\mathcal{L}$-valued extension of classic Horn clauses. Trying to design a corresponding $\mathcal{L}$-valued SLDNF resolution will be the challenge of this communication.

1 Motivation for extending Prolog

Our motivation for designing an $\mathcal{L}$-valued extension of a Prolog engine comes from practical considerations. In the context of multi-criteria decision aid methods, we want to compute choice recommendations [1] for a given decision-maker. Such recommendations rely heavily on a logical treatment of multiple and fuzzy information. If the first point is well served by logic programming techniques, the second point, i.e. fuzziness of information, is actually not basically supported by such techniques.

1.1 A didactic decision problem

We suppose that a decision-maker has to make a selection from a set A of three R&D projects represented as $A = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$. We furthermore assume that a given decision aid analysis has eventually conducted to the following out-ranking index on $A \times A$. General semantics of the out-ranking relation is given by "to be at least as good as".

1. **a** weakly out-ranks (credibility = 20%) **b**;
2. **a** most certainly out-ranks (credibility = 95%) **c**;
3. **b** out-ranks more or less (credibility $\cong$ 70%) **a**;
4. **b** most certainly out-ranks (credibility = 95%) **c**;
5. **c** out-ranks more or less (credibility $\cong$ 70%) **a** and **b**.

What R&D project should we recommend the decision-maker to choose ?

[1]The example we will discuss, is a variant of an example proposed by Bernard Roy at the 41th meeting of the EURO working group MCAD on "Multi-criteria Aid for Decisions" in Lausanne, March 1995 on the occasion of a lecture by Marc Roubens on the work about fuzzy kernels from Leonid Kitaïnik [11].

1.2 Solving this problem by using standard Prolog

Reasoning about preferences may let come up following relational concepts: two alternatives X and Y are *equivalent* if they outrank each other mutually and an alternative X *dominates* an alternative Y if X outranks Y but not the converse.

Written in Prolog, these concepts may be formulated as follows:

```
equivalent(X, Y) :-
      outranks(X, Y),
      outranks(Y, X).
dominates(X, Y):-
      outranks(X, Y),
      not(outranks(Y, X)).
```

The difficulty we are faced with is to represent the factual outranking relation from the above given outranking statements. Generally, a *cutting* technique like the following is used for this purpose: *to be considered true, the credibility of an outranking must be greater than 50%.* This gives for instance following (*true*) ground facts to consider:

```
outranks(a, c).
outranks(b, a).
outranks(b, c).
outranks(c, a).
outranks(c, b).
```

To try a ground query, let us ask whether a and c are equivalent? From the classical SLDNF resolution with left most selection criteria we get the following trace:

```
?- equivalent(a,c).       %% example run # 1
CALL   equivalent(a, c)
    CALL   outranks(a, c)     %% 1rst literal
    EXIT   outranks(a, c)
    CALL   outranks(c, a)     %% 2nd literal
    EXIT   outranks(c, a)
EXIT   equivalent(a, c)
yes
```

The query terminates successful as both required literal can be resolved with a true ground fact.

As second example, we consider now a general query involving the `dominates` predicate containing a negative literal. Following trace may be observed:

```
?- dominates(X, Y).           %% example run # 2
CALL   dominates(X, Y)
    CALL   outranks(X, Y)         %% 1rst literal
    EXIT   outranks(a, c)
    CALL   not outranks(c, a)     %% 2nd literal
        CALL   outranks(c, a)
        EXIT   outranks(c, a)
        CUT    not outranks(c, a)
    FAIL   not outranks(c, a)
    RETRY  outranks(X, Y)         %% 1rst literal
    EXIT   outranks(b, a)
    CALL   not outranks(a, b)     %% 2nd literal
        CALL   outranks(a, b)
        FAIL   outranks(a, b)
    EXIT   not outranks(a, b)
EXIT   dominates(b, a)
yes(X = b, Y = a) ;
CALL   fail
FAIL   fail
    RETRY  outranks(X, Y)         %% 1rst literal
    EXIT   outranks(b, c)
    CALL   not outranks(c, b)     %% 2nd literal
        CALL   outranks(c, b)
        EXIT   outranks(c, b)
        CUT    not outranks(c, b)
    FAIL   not outranks(c, b)
    RETRY  outranks(X, Y)         %% 1rst literal
    EXIT   outranks(c, a)
    CALL   not outranks(a, c)     %% 2nd literal
        CALL   outranks(a, c)
        EXIT   outranks(a, c)
        CUT    not outranks(a, c)
    FAIL   not outranks(a, c)
    RETRY  outranks(X, Y)         %% 1rst literal
    EXIT   outranks(c, b)
    CALL   not outranks(b, c)     %% 2nd literal
        CALL   outranks(b, c)
        EXIT   outranks(b, c)
        CUT    not outranks(b, c)
    FAIL   not outranks(b, c)
FAIL   dominates(c, b)
no (more) solutions
```

Only one clear dominance relation is observed between alternative **b** and **a**.

1.3 How "true" is a more or less credible fact ?

But what happens, if actual *truth* of the outranking relation requires at least a credibility of 75% ? This time the only *true* ground facts remaining are:

```
outranks(a, c).
outranks(b, c).
```

and the result of the same queries gives — first following trace:

```
?- equivalent(a, c)       %% example run # 3
CALL   equivalent(a, c)
    CALL   outranks(a, c)     %% 1rst literal
    EXIT   outranks(a, c)
    CALL   outranks(c, a)     %% 2nd literal
    FAIL   outranks(c, a)
    RETRY  outranks(a, c)     %% 1rst literal
    FAIL   outranks(a, c)
FAIL   equivalent(a, c)
no
```

— whereas the second predicate returns:

```
?- dominates(X, Y)            %% example run # 4
CALL   dominates(X, Y)
    CALL   outranks(X, Y)     %% 1rst literal
    EXIT   outranks(a, c)
    CALL   not outranks(c, a) %% 2nd literal
        CALL   outranks(c, a)
        FAIL   outranks(c, a)
    EXIT   not outranks(c, a)
EXIT   dominates(a, c)
yes(X = a, Y = c);
CALL   fail
FAIL   fail
    RETRY  outranks(X, Y)     %% 1rst literal
    EXIT   outranks(b, c)
    CALL   not outranks(c, b) %% 2nd literal
        CALL   outranks(c, b)
        FAIL   outranks(c, b)
    EXIT   not outranks(c, b)
EXIT   dominates(b, c)
yes(X = b, Y = c) ;
CALL   fail
FAIL   fail
no (more) solutions
```

This time, no equivalence is observed between the alternatives and two new dominance relations appear. Changing thus the credibility level for accepting the truth of a ground fact, changes drastically the outcome of the resolutions. If we accept for instance all outranking facts notwithstanding their credibility, we get all pairs of alternatives as equivalent and no alternative will dominate any other.

Choosing an adequate credibility level for accepting the ground facts therefore becomes a major methodological and practical problem. We would indeed be much better off if the logic computation could, from the beginning on and through all steps, take into account ground credibilities associated with the ground facts of the program.

1.4 The importance of negation by failure for efficient SLD resolution

Beside the practical problem above, we are also confronted with a computational problem. SLDNF resolution, through the *Negation as Failure* (NF) principle, makes the logical assumption that all not convincing outranking relations are to be considered as being false, i.e. absence of information results in negative information.

The operational need to follow this principle is well illustrated by the trace of example run #2 above and no implementation of a logic programming paradigm can bypass this principle without loosing a great deal of efficiency in computation.

Unfortunately, because of the the NF principle, the incidence of choosing different credibility levels for acceptance of the outranking statements, as seen in the traces of example runs #3 and #4 above, is more critical. Most careful choice of these cut levels becomes therefore crucial for implementing a fruitful decision aid. That's why we have great interest in being able to postpone as late as possible any necessity to de-fuzzify the initially given outranking statements.

Consequently we are looking for a possibility to integrate these credibilities coherently in our logical computations we want to run, returning the decision-maker more or less credible results. It is his eventually decision then to accept as true or not these results.

It is our goal in this paper to introduce a logical extension of classic Prolog resolution which allows to overcome these two major drawbacks.

2 From credibility to truthfulness of logical formulas

First, we introduce a clear semantic distinction between some underlying credibility calculus qualifying the truthfulness of given facts and clauses, and the subsequent acceptation of effective truthfulness of these logical expressions.

2.1 Basic credibility calculus

Definition 2.1. Let a represent an atomic formula or *atom* with which we may associate a finite rational degree of credibility $r(a) \in [0,1]$ expressing its potential truthfulness. If $r(a) = 1$, atom a is perceived as *certainly true*, and if $r(a) = 0$, atom a is perceived as *certainly false*. The complete ordered finite set of involved credibility values is denoted V. The underlying order is denoted $(V, \leq)$, where $\leq$ denotes a complete, reflexive, antisymmetric and transitive ordering.

Normally, degrees of credibility associated with a given atom a are expressed as 2-digits percentages. Certainly true atoms are 100% credible whereas certainly false atoms are 0% credible.

Definition 2.2. Let $(\mathcal{A}, r)$ be a set of atomic formulas a associated with corresponding degrees of credibility $r : \mathcal{A} \to V$. Let $\neg$, $\vee$, $\wedge$ and $\leftarrow$ denote respectively negation, disjunction, conjunction and implication. The set $\mathcal{E}$ of all *well formulated finite expressions* will be generated inductively from the following grammar:

$$\forall a \in \mathcal{A} \quad : \quad a \in \mathcal{E} \tag{1}$$

$$\forall x, y \in \mathcal{E} \quad : \quad \neg x \mid (x) \mid x \vee y \mid x \wedge y \mid x \leftarrow y \ \in \mathcal{E} \tag{2}$$

The unary operator $\neg$ has a higher precedence in the interpretation of a formula, but generally we use bracketing parentheses to control the application range of a given operator and thus to make all formulas have unambiguous semantics.

We extend the credibility calculus on such logical expressions in the following way:

Definition 2.3. Let $\mathcal{E}$ be a set of well formulated expressions. $\forall x, y \in \mathcal{E}$:

$$r(\neg x) = 1 - r(x) \tag{3}$$
$$r(x \vee y) = \max(r(x), r(y)) \tag{4}$$
$$r(x \wedge y) = \min(r(x), r(y)) \tag{5}$$
$$r(x \leftarrow y) = 1 \Leftrightarrow r(x) \geq r(y) \tag{6}$$

From the inductive definition of our well formulated expressions, we are able to compute credibility of such a formula in a logical evaluation domain $\mathcal{L} = (V, \leq, \max, \min, \neg, 0, \frac{1}{2}, 1)$. The classic *strong negation* operator $\neg$ acts as an anti-tonic bijection on V with the median value $\frac{1}{2}$ acting as negational fix-point. Classical *min* and *max* operators implement conjunctive respectively disjunctive credibilities with 0 and 1 as lower and upper limit. The implication operator $\leftarrow$ will be confined in the scope of this paper to certainly true denotation, therefore a necessary and sufficient condition for this credibility is provided in formula (6) above. Finally, we denote the couple $(\mathcal{E}, r)$ as $\mathcal{E}^{\mathcal{L}}$ and speak in the sequel simply of $\mathcal{L}$-valued expressions.

Two special such $\mathcal{L}$-domains are worthwhile noticing here: – first, the three-valued domain $\mathcal{L}_3 = (\{0, \frac{1}{2}, 1\}, \leq, \max, \min, \leftarrow, \neg, 0, \frac{1}{2}, 1)$ giving in fact a three-valued logic, and – secondly, the bi-valued (degenerated) domain $\mathcal{B} = (\{0, 1\}, \leq, \max, \min, \leftarrow, \neg, 0, 1)$ isomorphic to the classical Boolean logic.

Noticing the credibility of a given $\mathcal{L}$-valued expression, we are now able to induce its supposed truthfulness.

2.2 Defining the $\mathcal{L}$-truthfulness of an $\mathcal{L}$-valued expression

In Prolog as well as in classical bi-valued logic, it is general use to work syntactically only on the *truth* point of view of the logic, the *falseness* point of view being redundant through the coercion to the excluded middle. This argument allows to partly rely on the NF principle for SLD resolution. For instance, writing "relation(a, b)" means implicitly that we assume this atom being actually true and its negation false, otherwise, we would write "$\neg$ relation(a, b)". We are easily trapped by the "ontologic" property, the materiality of a formulation gives a fact and we insidiously induce truth from such formal existence.

Or, we will continue to rely syntactically on such an implicit truth point of view and always denote possible truthfulness induced from an underlying credibility calculus through a μ operator[2] acting as domain restriction on the credibility operator r. But we want nevertheless guarantee a coherent, i.e. symmetric treatment of affirmation and negation of truth and falseness.

[2]In fuzzy set theory, the μ operator generally denotes a fuzzy membership function. We choose here the same μ symbol on purpose as our main $\mathcal{L}$-valued formulas concern mostly such $\mathcal{L}$-valued characteristic functions

Definition 2.4. Let $x \in \mathcal{E}^{\mathcal{L}}$ be an $\mathcal{L}$-valued expression associated with credibility $r(x)$:

$$\mu(x) = r(x) \;\Leftrightarrow\; r(x) \geq r(\neg x)$$

Truthfulness of a given expression x is only defined in case the expression's credibility $r(x)$ exceeds the credibility $r(\neg x)$ of its negation $\neg x$. From our definition, it follows immediately that our induced $\mathcal{L}$-valued truth calculus is complete on every set $\mathcal{E}$ of well formulated $\mathcal{L}$-valued expressions in the sense that for each $x \in \mathcal{E}$: either $\mu(x)$ or $\mu(\neg x)$ is defined. We say for short that any expression $x \in \mathcal{E}^{\mathcal{L}}$ is *$\mathcal{L}$-true* or *$\mathcal{L}$-false*.

To illustrate our approach, let us now look at $\mathcal{L}$-valued truthfulness of certain classical tautologies or antilogies. For instance, for the tautology $(a \vee \neg a)$, we have $\mu(a \vee \neg a)$ always defined, as $\max(r(a), 1 - r(a)) \geq \frac{1}{2}$ in any case. Tautologies appear thus as being $\mathcal{L}$-true in any case. Therefore we call them $\mathcal{L}$-tautologies. On the other hand, for the antilogy $(a \wedge \neg a)$, we obtain the following truthfulness values: $\mu(a \wedge \neg a) = \frac{1}{2} \Leftrightarrow a = \neg a = \frac{1}{2}$. Truthfulness of this formula is undefined otherwise where as truthfulness of its contradiction, i.e. $\neg(a \wedge \neg a)$ is always defined, i.e. $\mathcal{L}$-true. Therefore, we will call such contradictory propositions $\mathcal{L}$-antilogies. As a main result of our construction, we recover all classical tautologies and antilogies as particular limit case $\mathcal{E}^{\mathcal{B}}$.

Indeed, let us investigate an implicative $\mathcal{L}$-tautology such as the *modus ponens* for instance. If we take a classical (Kleene-Dienes) definition of the implication, i.e. falseness of the conjunction of a and $\neg b$, we obtain that $\min(r(a), \max(1 - r(a), r(b))) \geq \frac{1}{2} \Rightarrow r(b) \geq \frac{1}{2}$. So that we obtain indeed the following $\mathcal{L}$-tautology: a and $a \Rightarrow b$ being conjointly $\mathcal{L}$-true always implies b being $\mathcal{L}$-true.

At this point, one may wonder if our $\mathcal{L}$-logic does not appear as an isomorphic structure to the classical bi-valued Boolean logic. But this is not the case, as the natural categorical limit $\mathcal{L}$-algebra is given by the $\mathcal{L}_3$ logic. But we may force the $\mathcal{L}$-valued truth calculus to take the bi-valued Boolean logic $\mathcal{B}$ as limit, if we exclude explicitly from the set of possible credibility values the negational fix-point, thus making it impossible to stay logically neutral in the underlying credibilities and hence also in our truthfulness assumptions.

The semantic difference between the logical fuzziness we introduce, and s-tandard fuzziness as modelled in fuzzy or multi-valued logic and/or in fuzzy set theory, remains – first in the operational distinction between the underlying propositional credibility calculus and the induced truthfulness calculus, and – secondly in the strict splitting of the credibilities into truthfulness supporting ones and in falsefulness supporting ones with a possible unique common intersection only in case of logical undeterminedness.

3 Logical Fuzzification and Polarization: an Adjoint Pair

In this section we investigate more categorically the previously introduced semantics through the general techniques of fuzzification and defuzzification. First we turn to logical fuzzification.

3.1 Introducing Logical Fuzzification

A common sense fuzzification of the classic bi-valued truth calculus consists in replacing the underlying bi-valued characteristic function with a multi-valued bounded real-valued characteristic function followed by the attempt to axiomatize such real-valued logical calculus in such a way to recover all known (and so generally accepted as useful) logical properties of the bi-valued case. In our opinion, this straightforward method suffers from two main drawbacks: – first, such simple multi-valued generalization of the bi-valued characteristic function is in contradiction with the necessarily split semantics of the implicitly modelled truth/falseness point of views and – secondly, some of the structural properties of a bi-valued logical framework appear as degenerated limit properties, impossible to observe in the general case without precisely questioning this generalization again. Our reason for thinking so is given, not so much by the axiomatic difficulties of the standard attempt, but rather more by the dubious operational properties of the corresponding standard defuzzification technique, denoted in the fuzzy literature as λ-cuts, where $\lambda \in V$ represents the level of credibility at which a given proposition is taken to be true or not.

To illustrate the difficulty, we consider the definition of cut relations as discussed in Fodor & Roubens (see [9, p.45]). Let us take a fuzzy outranking index, as shown in table (1). For a given $\lambda \in V$, Fodor & Roubens propose to consider the relation R_λ as being defined as the set of pairs $(a, b) \in A \times A$ such that $r(a, b) \geq \lambda$. What has escaped the authors' minds is the fact that a cut technique is necessarily either truth or falseness oriented and that the resulting crisp or bi-valued relation is, in their case, taken from the truth point of view. But this point of view cannot, in a standard logical frame, where a negational contradiction principle must exist, be separated from the complementary falseness point of view. Stating that for instance there exists a .20-cut relation on the index r as shown in Table (1) implicitly implies, in case we envisage a credibility calculus with a strong negation, an asymmetric treatment of truthfulness versus falsefulness. Indeed, we would consider proposition "$(a\,\mathrm{R}\,b)$" to be true, as $r(a, b) = .25 > .20$ and conjointly proposition "$(a \not\mathrm{R}\, b)$" to be also true as $1 - r(a, b) = .75 > .20$, and thereby we would introduce logical nonsense. But if we solely envisage λ-cut relations with $\lambda \geq \frac{1}{2}$, we may obtain possibly correct split truthfulness versus falsefulness semantics. Indeed, consider again the same pair $(a\,\mathrm{R}\,b)$. This time, we conclude that $(a \not\mathrm{R}\, b)$ as $r(a, b) = .25 < .50$, this result being coherent with the fact that $1 - r(a, b) = .75 > .50$.

Table 1: Example of Fuzzy Outranking Relation

r	a	b	c
a	-	.25	.95
b	.55	-	.95
c	.55	.55	-

Two lessons are to be learned from this example. First, every logical truth calculus conjointly addresses a mirrored falseness calculus through the necessary existence of a contradiction principle. Secondly, fuzzification models are not to be separated from intended defuzzification or logical polarization techniques, as defuzzification necessarily projects fuzziness into the split complementary semantics of truthfulness versus falsefulness.

The above considerations give rise to the following question: why does Boolean calculus nevertheless work, although it does not use such a split semantics?

In this specific case, the reason for this simplification lies in the operational restriction to two extreme truth values $\{0, 1\}$, with implied coercion to the excluded middle, which simplifies the underlying logical algebra to a point making the necessity to separate truthfulness values from falsefulness values disappear. Indeed, both points of view are completely determined one by the other, as no intermediate position is allowed. As soon as intermediate logical positioning is allowed, the necessity also appears to algebraically separate the truthfulness point of view from the falsefulness point of view and eventually to introduce a third term, the necessarily unique common truth/falseness point of view: logical undeterminedness. It appears from the algebraic framework that, in case we envisage a strong negational contradiction, this logical undeterminedness resides in the necessarily unique negational fix-point, i.e. the truthfulness/falsefulness value $\frac{1}{2}$ marking the common border line between appearing truthfulness versus appearing falsefulness (see Figure 1).

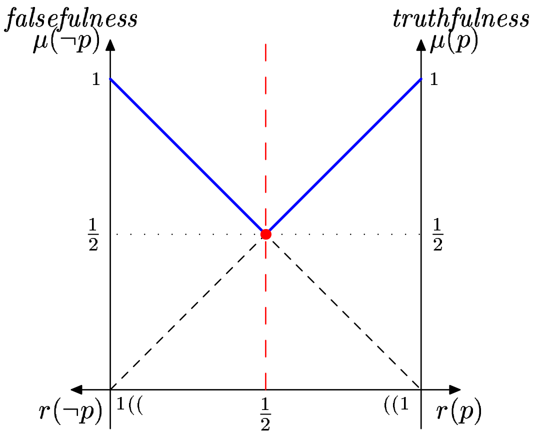

Figure 1: Split Truth/Falseness Semantics

To illustrate our approach, we consider a simple majority voting procedure generating atomic propositions with respective credibilities reflecting the positive or negative results of the individual voting. A proposition is true from the moment it gets more than half of the possible votes and it is false elsewhere. The exact result models the greater or lesser truthfulness or falsefulness of the result. But no proposition may be conjointly accepted and rejected unless it gets exactly half of the votes. In practice, this blocking case is some times avoided either by considering only odd numbers of possible voters or by a specific deblocking procedure like for instance counting the chairman's vote for double in this situation.

One may question the simple majority rule. Why is it just half and not lesser of the votes that distribute truthfulness and falsefulness? Mostly for practical reasons: if less than $\frac{1}{2}$ of positive votes would be sufficient to accept a proposition, a proposition could in practice be both accepted and rejected, which would produce practical (logical) nonsense. On the contrary, higher qualified majority rules are very often used in practice and especially for important decisions. How can we cope with this issue in our logical framework?

3.2 On Natural Logical Polarization

What we are looking for is a defuzzification or logical polarization operator denoted π, similar to the λ-cut but compatible with our split truth/falseness semantics.

In Bisdorff & Roubens [5], we have for the first time introduced such a polarization operator π_β, which we named the β-cut[3].

Definition 3.1. Let $\mathcal{E}^{\mathcal{L}}$ be a set of logical expressions associated with credibility $r : \mathcal{E}^{\mathcal{L}} \to V$, where $\mathcal{L} = \{V, \geq, \neg, \min, \max, \leftarrow, 0, \frac{1}{2}, 1\}$ underlies algebraically the propositional credibility calculus. Let π_β represent a logical polarization operator $\pi_\beta : \mathcal{E}^{\mathcal{L}} \to \mathcal{E}^{\mathcal{L}_3}$ defined as follows: $\forall x \in \mathcal{E}^{\mathcal{L}}$ and $\forall \beta \in]\frac{1}{2}, 1]$:

$$\pi_\beta(r(x)) = \begin{cases} 1 & \Leftrightarrow \quad r(x) > \beta \\ 0 & \Leftrightarrow \quad r(x) < 1 - \beta \\ \frac{1}{2} & \Leftrightarrow \quad 1 - \beta \leq r(x) \leq \beta \end{cases}$$

That this polarization operator π_β indeed satisfies our formal expectations may be summarized by stating the following theorem:

Theorem 3.1. *Let $\mathcal{E}^{\mathcal{L}}$ be a set of $\mathcal{L}$-valued expressions. Let μ denote our truthfulness operator, π_β the β-cut polarization operator and $\pi_{\frac{1}{2}}$ the median β-cut operator. The following categorical equations are verified:*

$$\mu \circ \pi_{\frac{1}{2}} = \pi_{\frac{1}{2}} \circ \mu; \tag{7}$$
$$\mu \circ \pi_\beta \supseteq \pi_\beta \circ \mu. \tag{8}$$

Indeed, considering equation (7), we must show that the $\pi_{\frac{1}{2}}$ operation is a natural transformation of $\mathcal{L}$-valued expressions with respect to our μ operator. Let us first consider disjunction (respectively conjunction) of two $\mathcal{L}$-valued expressions $x, y \in \mathcal{E}^{\mathcal{L}}$. Let $\pi_{\frac{1}{2}}(x)$ and $\pi_{\frac{1}{2}}(y)$ be the associated $\frac{1}{2}$-cut propositions. $\pi_{\frac{1}{2}}(\mu(x \vee / \wedge y)) = \max / \min(r(x), r(y)) \Leftrightarrow (r(x) \geq \frac{1}{2}) \vee / \wedge (y \geq \frac{1}{2}) \Leftrightarrow \mu(\pi_{\frac{1}{2}}(x) \vee \pi_{\frac{1}{2}}(x)) = \max / \min(r(x), r(y))$. The same straightforward argument applies to the two other logical operators in $\mathcal{L}$, i.e. strong negation and our ordinal implication. Considering now equation (8), it is precisely this last ordinal implication that appears as more restrictive than a negative $\mathcal{L}$-valued Kleene-Dienes implication, thus generally making the π_β operator only semi-natural for the μ transform (see Bisdorff & Roubens [5]). $\square$

It is clear that the demonstration above is highly dependent on the choice of the $\mathcal{L}$-algebra and it would be interesting to furthermore characterize naturally defuzzifiable credibility calculi in the sense above.

Let us now turn to logic programs. What we need here is an $\mathcal{L}$-valued extension of SLDNF resolution which turns out to be natural with respect to our β-cutting technique.

[3]The name we gave this logical polarization operator is derived from the fact that the standard λ-cut is sometimes also named α-cut.

4 $\mathcal{L}$-valued logic programs

In this section we first delimit our definition of logic programs. In a second part we apply our credibility calculus to such logic programs and we extend SLDNF resolution to such programs. The rest of this section illustrates these ideas with the help of our introductory examples.

4.1 Syntax

In the scope of this discussion, we restrict our attention to such atoms a which may be represented as n-ary relational predicates $p(t_1, ..., t_n)$ among terms, n being the arity of the predicate p. Furthermore we denote by X, Y and Z the place-holding variables appearing in such atoms. An atom a is called *ground* if no variable occurs in it.

A *literal* is an atom a or its negation $\neg a$ and is denoted by the letter l.

Definition 4.1. An *implying clause* is a formula $c \in \mathcal{E}$ of the form:

$$a \leftarrow l_1 \wedge ... \wedge l_m \tag{9}$$

where a is the *head* atom; $l_1 \wedge ... \wedge l_m$, the body of the clause are literals and $m \geq 1$.

Implying clauses allow to implement implicative tautologies under the form of modus ponens inference schemes. In our construction we introduce furthermore tautological equivalence schemes in order to correctly catch declarative semantics of truth and falseness of facts and definitions.

Definition 4.2. A *defining clause* is a formula $c \in \mathcal{E}$ of the form:

$$a \leftrightarrow l_1 \wedge ... \wedge l_n \tag{10}$$

where the $\leftrightarrow$ corresponds to a two-way $\leftarrow$-implication and the number n of literals in the body is ≥ 0. If the body of a defining clause is empty, i.e. $n = 0$, we speak of a *fact* and if the head is empty, we speak of a *goal*.

Definition 4.3. We call *logic program*, denoted P, a finite set of facts, defining and implying clauses. The set of facts is called the ground of P and is denoted P^g. The set of defining (respectively implying) clauses of P is denoted as P^d (respectively P^i).

We now extend our credibility calculus to such logic programs.

4.2 $\mathcal{L}$-valued logic programs

Definition 4.4. Let $P = (P^g, P^i, P^d)$ be a logic program.

$$\begin{aligned}
\forall a \in P^g \quad &: \quad r(a) \in V && (11)\\
\forall c \in (P^d \cup P^i) \quad &: \quad r(c) = 1 && (12)\\
\forall (a \leftarrow l_1 \wedge ... \wedge l_m) \in P^i \quad &: \quad r(a) = \max(\frac{1}{2}, r(l_1 \wedge ... \wedge l_m) && (13)\\
\forall (a \leftrightarrow l_1 \wedge ... \wedge l_m) \in P^d \quad &: \quad r(a) = r(l_1 \wedge ... \wedge l_m) && (14)\\
& && (15)
\end{aligned}$$

All ground facts P^g may be associated with any credibility from V, whereas defining and implying clauses are restricted to be considered certainly true formulas.

Contrary to the Boolean case, we cannot rely on a negative (Kleene-Dienes) definition of implication, i.e. to infer from the credibility of clause's body, that of its head. Indeed, in Boolean logic, the falseness of the antecedent implies, through negational complementarity, the consequent's default truthfulness (in fact a double negation as success principle). We rather prefer in this case to associate the consequent with an undetermined logical denotation.

For defining clauses, as they implement tautological equivalences, we simply give the head, the body's global credibility. In this case indeed, if the body is rather true (respectively false), the head will be also rather true (respectively false). Notice that a recursive clause may nether be a defining clause as the recursive part of the definition must be completed by a bottom (initial or terminal) part.

In order to determine the credibility associated with an individual literal of the body of a clause, we propagate credibilities through standard SLD resolution. Analogue to the NF strategy, we base our resolution strategy on an ***$\mathcal{L}$-Untruth as Failure*** principle and we denote this resolution consequently as SLD$\mathcal{L}$-UF resolution.

Definition 4.5. The SLD$\mathcal{L}$-UF resolution is based on following principles:

1. Resolution of the literals follows the left most literal selection principle.
2. For implying clauses we fail resolution as soon as the currently resolved literal gets an $\mathcal{L}$-untrue credibility. Failure transfers the $\mathcal{L}$-undeterminedness to the clause's head.
3. For defining clauses we exhaustively resolve the list of literals of the body and return the body's resulting credibility ($\mathcal{L}$-true or $\mathcal{L}$-untrue) to the head.

Let us first illustrate our resolution strategy on ground queries through defining clauses.

4.3 SLD$\mathcal{L}$-UF resolution of defining clauses

In a 2-digits percentage credibility evaluation domain, the whole set of ground facts may be annotated as follows:

```
outranks(a, b){25}
outranks(a, c){95}
outranks(b, a){55}
outranks(b, c){95}
outranks(c, a){55}
outranks(c, b){55}
```

We use {..} to indicate credibilities attached to a fact or a clause. Place-holding credibility variables are denoted C or $C_1, C_2, \ldots$. To distinguish defining clauses from implying clauses, we use a two-way Prolog body separator "`-:-`".

Both relational concepts, i.e. equivalence and dominance may thus be expressed as defining clauses:

```
equivalent(X, Y){min(C_1, C_2)}  -:-
      outranks(X, Y){C_1},                   %% 1rst literal
      outranks(Y, X){C_2}.                   %% 2nd literal
dominates(X, Y){min(C_1, C_2})}  -:-
      outranks(X, Y){C_1},                   %% 1rst literal
      not( outranks(Y, X){1 - C_2} ){C_2}. %% 2nd literal
```

When executing ground queries similar to example run #1 above, we obtain through SLD$\mathcal{L}$-UF resolution following traces:

```
?- equivalent(a, c){C}.        %% example run # 5 : ground query
CALL   equivalent(a, c){C}
    CALL   outranks(a, c){C_1}               %% 1rst literal
    EXIT   outranks(a, c){95}
    CALL   outranks(c, a){C_2}               %% 2nd literal
    EXIT   outranks(c, a){55}
EXIT   equivalent(a, c){55}
{C = 55}.
?- equivalent(a, b){C}.
CALL   equivalent(a, b){C}
    CALL   outranks(a, b){C_1}               %% 1rst literal
    EXIT   outranks(a, b){25}
    CALL   outranks(b, a){C_2}               %% 2nd literal
    EXIT   outranks(b, a){55}
EXIT   equivalent(a, c){25}
{C = 25}.
```

In the cases of example run #5, our resolution strategy returns as $\mathcal{L}$-true the equivalence between **a** and **c** (with credibility 55%) and as $\mathcal{L}$-false the equivalence between **a** and **b** (with credibility 1 - 25 = 75%) . If we think that the credibilities of the results must exceed for instance 70% in order to be considered as reliable, i.e. certainly true, i.e. we apply a 70%-cut to the results, we remain solely with the $\mathcal{L}$-false statement as the first one becomes $\mathcal{L}$-undetermined.

Let us now look for query concerning a non ground instance of a defining clause.

```
?- dominates(X, Y){C}. %% example run # 6 : non ground query
CALL   dominates(X, Y){C}
    CALL   outranks(X, Y){C_1}                  %% 1rst literal
    EXIT   outranks(a, b){25}
    CALL   not outranks(b, a){C_2}              %% 2nd literal
        CALL   outranks(b, a){1 - C_2}
        EXIT   outranks(b, a){25}
    EXIT   not outranks(b, a){75}
EXIT   dominates{a, b){25}
(X = a, Y = b){C = 25};
RETRY fail
FAIL  fail
    RETRY  outranks(X, Y){C_1}                  %% 1rst literal
    EXIT   outranks(a, c){95}
    CALL   not outranks(c, a){C_2}              %% 2nd literal
```

```
        CALL   outranks(c, a){1 - C_2}
        EXIT   outranks(c, a){55}
    EXIT   not outranks(c, a){45}
EXIT   dominates(a, c){45}
(X = a, Y = c){C = 45};
CALL   fail
FAIL   fail
    RETRY  outranks(X, Y){C_1}                %% 1rst literal
    EXIT   outranks(b, a){55}
    CALL   not outranks(a, b){C_2}            %% 2nd literal
        CALL   outranks(a, b){1 - C_2}
        EXIT   outranks(a, b){25}
    EXIT   not outranks(a, b){75}
EXIT   dominates(b, a){55}
(X = a, Y = c){C = 55};
CALL   fail
FAIL   fail
...
...
(X = c, Y = b){C = 5};
CALL   fail
FAIL   fail
EXIT   dominates(X, Y){50}
(X = X, Y = Y){C = 50}.
```

Example run #6 collects all possible combinations of instantiable facts with their corresponding credibilities. A "no trace" run shows the complete list of results:

```
?- dominates(X, Y){C}
(X = a, Y = b) {25};
(X = a, Y = c) {45};
(X = b, Y = a) {55};
(X = b, Y = c) {45};
(X = c, Y = a) {5} ;
(X = c, Y = b) {5} ;
(X = X, Y = Y) {50}.
```

Applying defuzzification operator μ on this list, we obtain the same results as for example run #2. Indeed, solely ground fact "`dominates(b, a)`" is $\mathcal{L}$-true with a credibility of 55%. All other computed facts are $\mathcal{L}$-false. Notice the last undetermined result. This means that whenever we try to query a fact not being part of the Herbrand base of the program in question, we get a general $\mathcal{L}$-undetermined result.

Indeed, let us finally query a non existing ground fact:

```
?- equivalent(e, f){C}.        %% example run # 7
CALL   equivalent(e, f){C}
    CALL   outranks(e, f){C}          %% 1rst literal
    FAIL   outranks(e, f){50}
FAIL  equivalent(e, f){50}
{C = 50}.
```

Contrary to standard SLDNF resolution, general failure does not produce a negative answer, but an $\mathcal{L}$-undetermined one. We may not conclude from the absence of information about hypothetical alternatives **e** and **f** that the outranking relation does not exist between them but rather more we infer that we don't know if this outranking statement is to be considered true or false, so that we qualify this potential ground fact as logically undetermined.

Let us now turn to a more complicated case.

4.4 SLD$\mathcal{L}$-UF resolution of implying clauses

As an example, we investigate the $\mathcal{L}$-valued computation of following recursive definition.

```
path_connected(X, Y, _, _){C}   :-
               outranks(X, Y){C}.
path_connected(X, Y, N, Nmax){min(C_1, C_2)} :-
               N < Nmax,
               outranks(X, Z){C_1},
               N1 is N + 1,
               path_connected(Z, Y, N1, Nmax){C_2}.
```

Two alternatives X and Y are path-connected if either, they are directly related through a ground outranking relation, or there exists a third alternative Z such that X and Z are directly related and there exists a path-connected relation between Z and Y of length $N < Nmax$. Executing the following ground query gives the trace below:

```
?- path_connected(a, b, 1, 2){C}.          %% example run # 7
CALL   path_connected(a, b, 1, 2){C}       %% implying clause
    CALL   outranks(a, b){C_1}             %% defining clause
EXIT   outranks(a, b){25}
FAIL   path_connected(a, b, 1, 2){50}
RETRY  path_connected(a, b, 1, 2){C}       %% implying clause
    CALL   1 < 2
    EXIT   1 < 2
    CALL   outranks(a, Z){C_1}             %% defining clause
    EXIT   outranks(a, c){95}
    CALL   N2 is 1 + 1
    EXIT   2 is 1 + 1
    CALL   path_connected(c, b, 2, 2) {C_2} %% implying clause
           CALL   outranks(c, b) {C_2_1}    %% defining clause
           EXIT   outranks(c, b) {55}
    EXIT   path_connected(c, b, 2, 2) {55}
EXIT   path_connected(a, b, 1, 2){55}      %% implying clause
{C = 55}.
```

In this example run #7, we apply our *$\mathcal{L}$-Untruth as Failure* principle in the same sense as is implemented the NF principle, and as expected, we get an $\mathcal{L}$-valued result that is, through our μ defuzzification, identical with the crisp bi-valued SLDNF resolution.

Finally we try in last section to explore general semantics of $\mathcal{L}$-valued logic programs.

5 A closer look at declarative $\mathcal{L}$-interpretations

In this last section, we start by putting up a convenient terminology about interpretations of logic programs (inspired by [1, 10, 3]). In a second part, we show that SLD$\mathcal{L}$-UF resolution is a natural extension of SLDNF resolution. Finally, in a last part, we generalize this result to Herbrand models of $\mathcal{L}$-valued logic programs.

5.1 $\mathcal{L}$-valued interpretations

Definition 5.1. The *Herbrand base* $\mathcal{H}(P)$ of a logic program P is the set of all variable-free terms, called facts and denoted a, that may be formed from the constants and relational predicates appearing in P. By convention we add to this set a general undetermined term ϵ capturing all possible terms outside of P, i.e. constants and relational predicates.

Proposition 5.1. *Let $P^{\mathcal{L}}$ be an $\mathcal{L}$-valued logic program and let $P_{/>\frac{1}{2}}$ be the median β-cut restricted program, where we remove all $\mathcal{L}$-untrue ground facts from $P^{\mathcal{L}}$.*

$$\mathcal{H}(P_{/>\frac{1}{2}}) \subseteq \mathcal{H}(P^{\mathcal{L}}) \tag{16}$$

Indeed, as $P_{/>\frac{1}{2}}$ is a subset of $P^{\mathcal{L}}$, the difference between both bases is essentially given by the subse of variable-free terms which may be formulated with potential $\mathcal{L}$-false ground facts appearing in $P^{\mathcal{L}}$. $\square$

Let us now define interpretations on Herbrand bases.

Definition 5.2. Let P be an $\mathcal{L}$-valued logic program and let I be an $\mathcal{L}$-valuation of $\mathcal{H}(P)$, i.e. $I : \mathcal{H}(P) \to V$ with $\forall a \in \mathcal{H}(P)$, $I(a)$ giving the credibility associated with fact a. Such $\mathcal{L}$-valuations I will be called $\mathcal{L}$-interpretations. We denote $\mathcal{I}^{\mathcal{L}}$ the set of all possible $\mathcal{L}$-interpretations we may associate with a given Herbrand base. The trivial, all $\frac{1}{2}$-valued interpretation is denoted M.

We may define on $\mathcal{I}^{\mathcal{L}}$ a pair of dual operators '$\oplus$' and '$\ominus$':

Definition 5.3. Let $I, J \in \mathcal{I}^{\mathcal{L}} : \forall a \in \mathcal{H}(P)$:

$$(I \oplus | \ominus J)(a) = \begin{cases} \max|\min(I(a), J(a) & \Leftrightarrow \quad I(a) \geqq \frac{1}{2} \wedge J(a) \geqq \frac{1}{2}, \\ \min|\max(I(a), J(a) & \Leftrightarrow \quad I(a) \leqq \frac{1}{2}, \wedge J(a) \leqq \frac{1}{2}, \\ \frac{1}{2} \quad \text{elsewhere.} \end{cases}$$

These "additive" operators are linked to a sharpness ordering (cf. Bisdorff & Roubens [5]) of $\mathcal{L}$-interpretations in the following way:

Definition 5.4. Let $I, J \in \mathcal{I}^{\mathcal{L}}$ be two $\mathcal{L}$-interpretations. We say that I is *sharper* than J, noted $I \succcurlyeq J$ iff $\forall a \in \mathcal{H}(P)$: either $(I(a) \leqq J(a) \leqq \frac{1}{2})$ or $\frac{1}{2} \leqq J(a) \leqq I(a))$.

The sharpness relation '$\succcurlyeq$' on the set $\mathcal{I}^{\mathcal{L}}$ of all interpretations defined on a given Herbrand base $\mathcal{H}(P)$, gives a partial order $(\mathcal{I}^{\mathcal{L}}, \succcurlyeq)$ with the trivial median-valued interpretation M as unique minimum element and all possible $\mathcal{B}$-valued (crisp) interpretations as the set of maximal (sharpest) elements.

Proposition 5.2. *Let $I, J \in \mathcal{I}^{\mathcal{L}}$ be two $\mathcal{L}$-valued interpretations defined on a Herbrand base $\mathcal{H}(P)$.*

$$I \succcurlyeq J \quad \Rightarrow \quad (I \oplus J \succcurlyeq I) \wedge (I \oplus J \succcurlyeq J), \tag{1}$$
$$I \succcurlyeq J \quad \Rightarrow \quad (I \succcurlyeq I \ominus J) \wedge (J \succcurlyeq I \ominus J). \tag{2}$$

Indeed, $\oplus$-addition (respectively $\ominus$-subtraction) of two $\succcurlyeq$-comparable interpretations gives an overall sharper (respectively fuzzier) interpretation as result. This is an immediate consequence of definition 5.4 and definition 5.3. $\square$

But we may more precisely isolate the subset of $\mathcal{L}$-valued interpretations which will give a $\oplus$-$\ominus$-lattice.

Definition 5.5. Let $I, J \in \mathcal{I}^{\mathcal{L}}$ be two $\mathcal{L}$-valued interpretations defined on a Herbrand base $\mathcal{H}(P)$. We say that I and J are **non-contradictory** iff $\forall a \in \mathcal{H}(P) : \big(I(a) \leqq \frac{1}{2} \Rightarrow J(a) \leqq \frac{1}{2}\big) \wedge \big(\frac{1}{2} \leqq I(a) \Rightarrow \frac{1}{2} \leqq J(a)\big)$. We shall note this relation as $I \cong J$.

Or "non-contradiction" is precisely the restriction functor on $\mathcal{L}$-valued interpretations we look for.

Proposition 5.3. *Let $I, J \in \mathcal{I}^{\mathcal{L}}$ be two $\mathcal{L}$-valued interpretations defined on a Herbrand base $\mathcal{H}(P)$.*

$$I \cong J \quad \Leftrightarrow \quad (I \oplus J \succcurlyeq I) \wedge (I \oplus J \succcurlyeq J). \tag{1}$$
$$I \cong J \quad \Leftrightarrow \quad (I \succcurlyeq I \ominus J) \wedge (J \succcurlyeq I \ominus J). \tag{2}$$

Verification of these properties is straight forward. $\square$

But then, we may notice that these "additive" operators $\oplus$ and $\ominus$ define in fact on a given $\cong$-congruence class of $\mathcal{I}^{\mathcal{L}}$, a lattice structure with its associated median β-cut interpretation as top element and the trivial $\mathcal{L}$-undetermined interpretation M as bottom element.

Proposition 5.4. *Let $I \in \mathcal{I}^{\mathcal{L}}$ be an interpretation defined on a Herbrand base $\mathcal{H}(P)$. Let $\mathcal{I}^{\mathcal{L}}_{/\cong I}$ be the "non-contradictory" congruence class associated with this interpretation I. If $M \in \mathcal{I}^{\mathcal{L}}$ is the trivial all median-valued interpretation, and $I_{>\frac{1}{2}}$ its associated median β-cut interpretation, the algebraic structure $\big(\{\mathcal{I}^{\mathcal{L}}, \oplus, \ominus, M, I_{>\frac{1}{2}}\big)$ gives a distributive lattice with relation M as bottom and $I_{>\frac{1}{2}}$ as top element.*

The $\oplus$-operator acts as lattice-join and the $\ominus$-operator as lattice-meet in a product of symmetric median-folded $\mathcal{L}$-algebras restricted to non-contradictory interpretations. The $\ominus$-operator has a lower limit which is the trivial all-median-valued interpretation M, whereas the $\oplus$-operator has an upper limit respective in the median β-cut interpretation. Double distributivity of $\oplus$ and $\ominus$ easily follows from the distributivity of the max and min operators of the underlying truth calculus algebra [4]. $\square$

[4]The $\oplus, \ominus$-structure is not directly a boolean algebra (a complemented distributive lattice), as the complement of a given interpretation is in general not unique. We denote these structures as "projectively" boolean.

We are now equipped to compare corresponding interpretations from SLD$\mathcal{L}$-UF and SLDNF resolutions.

5.2 SLD$\mathcal{L}$-UF resolution: a natural extension of SLDNF resolution

It is easily seen that SLD resolutions of a query Q from a program P construct in fact interpretations I on $\mathcal{H}(P)$.

In the classic Prolog case, SLDNF resolution constructs interpretations that are evaluated in a $\{YES, FAIL, NO\}$-valued credibility domain which we may identify with an $\mathcal{L}_3$ domain. For our purpose we denote all $FAIL$ and NO results as $\mathcal{L}$-untrue instantiations of the query. In the case of general $\mathcal{L}$-valued programs, SLD$\mathcal{L}$-UF resolutions construct interpretations that are evaluated in $\mathcal{L}$.

Proposition 5.5. *Let $P^{\mathcal{L}}$ be an $\mathcal{L}$-valued logic program and let $P_{/>\frac{1}{2}}$ be the median β-cut restricted program corresponding to $P^{\mathcal{L}}$. Let Q be the same query addressed to both programs. We denote $I(Q)$ the interpretation constructed through SLD$\mathcal{L}$-UF resolution of Q on $P^{\mathcal{L}}$ and we denote $J(Q)$ the interpretation constructed through SLDNF resolution of Q on $P_{/>\frac{1}{2}}$. Let M denote the trivial $\frac{1}{2}$-valued interpretation. Then we have:*

$$J(Q) \succcurlyeq I(Q) \succcurlyeq M \tag{3}$$

Indeed, definition (4.5) of SLD$\mathcal{L}$-UF resolution gives us the key to this result.

Let us first consider a query Q_ϵ formulated outside the Herbrand terms in $P_{/>\frac{1}{2}}$. Failure of Q_ϵ will be returned in both resolutions so that $J(Q_\epsilon) = I(Q_\epsilon) = M$.

Let Q_g be a closed atomic query on a clause contained in the Herbrand base $\mathcal{H}(P_{/>\frac{1}{2}})$. If SLDNF is successful, so will be SLD$\mathcal{L}$-UF with $J(Q_g) \geq I(Q_g) > \frac{1}{2}$. If SLDNF fails, either SLD$\mathcal{L}$-UF succeeds with $I(Q_g) \leq \frac{1}{2}$ in case the clause is a defining one or equally fails with $I(Q_g) = \frac{1}{2}$.

More generally, let Q_x be a generic query on a defining clause. SLDNF resolution will collect all possible successful instantiations denoted as $inst_J(Q_x)$. Again SLD$\mathcal{L}$-UF resolution, in case Q_x is a defining clause, will collect al-l ground instantiations in a set $inst_I(Q_x)$. Or, $inst_J(Q_x) \subseteq inst_I(Q_x)$ and $\forall a \in inst_J(Q_x) : J(a) \geq I(a) > \frac{1}{2}$.

Finally, let Q_i be a query concerning an implying clause. For such clauses, SLDNF and SLD$\mathcal{L}$-UF resolutions produce the same run traces and successful instantiations, i.e. $inst_J(Q_i) = inst_I(Q_i)$ and again $\forall a \in inst_J(Q_i) : J(a) \geq I(a) > \frac{1}{2}$. $\square$

As a corollary of this proposition we obtain thus that SLD$\mathcal{L}$-UF and SLDNF resolution of the same query Q on $P^{\mathcal{L}}$ respectively $P_{/>\frac{1}{2}}$ return non-contradictory interpretations, i.e. $\mathcal{L}$-interpretations belonging to a same congruence class $I^{\mathcal{L}}_{/\cong I(Q)}$, which is determined by interpretation $I(Q)$.

SLD$\mathcal{L}$-UF appears through this result as a natural $\mathcal{L}$-valued extension of SLDNF resolution and we can extend this result to $\mathcal{L}$-valued Herbrand models of a program.

5.3 $\mathcal{L}$-valued Herbrand models

Following the classical Van Emden and Kowalski [14] way, we define now a immediate consequence operator T_P on the set of possible Herbrand interpretations $\mathcal{I}^{\mathcal{L}}$ we may associate to a given program $P^{\mathcal{L}}$.

Every possible finite query Q on P requires a finite number n of execution steps for resolution. We denote $\mathcal{Q}^n$ the set of queries taking n steps for ground resolution. $\mathcal{Q}$ may then be partitioned as follows:

$$\mathcal{Q} = \bigcup_{i=1}^{n} \mathcal{Q}^i \tag{4}$$

We now define inductively the classic closure operator T_P on the base of the SLD$\mathcal{L}$-UF resolution.

Definition 5.6.

$$\begin{array}{rcl} T_P(0) & = & M \\ T_P(i) & = & \bigoplus\limits_{Q}\{I(Q) : Q \in \mathcal{Q}^i\}, i = 1 \ldots n \end{array}$$

In the recursive step above, $I(Q)$ gives the $\mathcal{L}$-interpretation associated with the SLD$\mathcal{L}$-UF resolution of query Q and each $T_P(i)$ "sums up" interpretations of all possible queries of resolution length i in P.

Taking powers of the T_P operator allows then to compute the $\mathcal{L}$-valued Herbrand model of such a program.

Definition 5.7.

$$\begin{array}{rcl} T_P{\uparrow}(0) & = & T_P(0) \\ T_P{\uparrow}(i) & = & T_P{\uparrow}(i-1) \oplus T_P(i), i = 1 \ldots n \end{array}$$

If we restrict our attention to logic programs executing all possible queries in a finite number of resolution steps, the closure operator T_P is *finitary* such that there necessarily exists for T_P a prefixed interpretation which we denote as $T_P{\uparrow}^{\omega}$.

From the definition of the $\oplus$-operator and from the fact that $\mathcal{Q}$ is partitionned by the length of the resolution, it follows that the T_P operator is necessarily *growing*, so that consequently, the prefixed interpretation $T_P{\uparrow}^{\omega}$ becomes a unique least interpretational fix-point which we identify with the least $\mathcal{L}$-valued *Herbrand model* denoted $\mathcal{M}(P^{\mathcal{L}})$.

Proposition 5.6.

$$\mathcal{M}(P^{\mathcal{L}}) \ = \ T_P{\uparrow}^{\omega} \tag{5}$$

We may generalize now the previous result concerning naturality of our $\mathcal{L}$-extension to such $\mathcal{L}$-valued Herbrand models.

Theorem 5.1. *Let $\mathcal{P}^{\mathcal{L}}$ denote the class of $\mathcal{L}$-valued programs, let $\mathcal{M}^{\mathcal{L}}$ denote the class of corresponding Herbrand models and let $\pi_{\frac{1}{2}}$ denote the median β-cut operator. Let $\mathcal{P}^{\mathcal{L}_3}$ denote the class of $\mathcal{L}_3$-valued programs and $\mathcal{M}^{\mathcal{L}_3}$ their associated Herbrand models.*

The following diagram then commutes:

$$\begin{tikzcd}
\mathcal{P}^{\mathcal{L}} \arrow[r, "\pi_{\frac{1}{2}}"] \arrow[d, "T_P"] & \mathcal{P}^{\mathcal{L}_3} \arrow[d, "T_P"] \\
\mathcal{M}^{\mathcal{L}} \arrow[r, "\pi_{\frac{1}{2}}"] & \mathcal{M}^{\mathcal{L}_3}
\end{tikzcd}$$

The theorem appears as corollary of theorem (3.1) and of proposition (5.5) above. $\square$

This all important theorem is a major consequence of the naturality of the β-cut technique as shown in Theorem (3.1) above and it allows us to coherently extend standard Prolog programs and SLDNF resolution to a multi-valued logical truth calculus. We may thus either choose to defuzzify the ground facts and compute in standard Prolog, or compute directly the fuzzy ground facts and de-fuzzify the results we obtain herewith. The outcome will be the same.

Conclusion

In this paper we have introduced a multi-valued extension of standard Prolog programs. To the classic SLD resolution with the Negation as Failure principle we have associated an SLD resolution with a corresponding Undeterminedness as Failure principle which allows us to coherently extend standard Prolog computation to this multi-valued universe.

Thus our major motivation we exposed in the first section of this paper, i.e. looking for a possibility to coherently integrate credibilities of ground facts in our logical computations without loosing the efficiency of the negation as failure principle in implementing the SLD resolution has received an optimistic answer, – at least from the theoretically feasibility point of view.

From a practical point of view this result gives us the theoretical possibility to postpone any de-fuzzification of facts to the instatiated results. It is then the actual decision-maker's last and definite responsibility to qualify as *true or not* the more or less credible results.

It remains to realize an efficient implementation of such an $\mathcal{L}$-Prolog engine so as to verify that we are effectively able to solve not only tiny but also real-sized problems such as we need to solve with standard Prolog engines. So, much enjoyment appears ahead for enthusiastic and courageous programmers.

Finally, I would like to thank Krzysztof Apt for his kindly insisting to invite me at the 9th Benelog workshop in Amsterdam. This motivation gave me the occasion to formulate my intuition.

References

[1] Apt, K.R., Gabbrielli, M. and Pedreschi, D., A closer look at declarative interpretations, *J. Logic Programming*, 28(2):147-180,(1996)

[2] Apt, K.R. and Doets, K., A new definition of SLDNF-resolution, *J. Logic Programming*, 18(2):177-190,(1994)

[3] Apt, K.R., Blair, H.A. and Walker, A., Towards a theory of declarative knowledge, *Foundations of deductive databases and Logic Proamming* (L. Minker, Ed.), Morgan Kaufmann Publishers, Los Altos, CA, 89-148, 1988

[4] Bisdorff, R., Logical foundation of fuzzy preferences with application to Electre decision aid methods, *European Journal of Operational Research*, special issue from EURO XV/INFORMS XXXIV Conference (Barcelona, 1997) (Valares-Tavares, L. and Dean, B.V. Ed.), forthcoming

[5] Bisdorff, R. and Roubens, M., On defining fuzzy kernels from $\mathcal{L}$-valued simple graphs, In *Proceedings of the conference IPMU'96*, pages 593-599, Granada, July 1996

[6] Bisdorff, R. and Roubens, M., On defining and computing fuzzy kernels from $\mathcal{L}$-valued simple graphs. In Da Ruan et al. (ed.), *Intelligent Systems and Soft Computing for Nuclear Science and Industry, FLINS'96 workshop*, pages 113–123, World Scientific Publishers, Singapoure, 1996

[7] Bisdorff, R.,On computing kernels from $\mathcal{L}$-valued simple graphs. In *Proceedings of the conference EUFIT'97*, volume 1, pages 97–103, Aachen, September 1997

[8] De Baets, B., Van de Walle, B. and Kerre, E.E., Fuzzy preference structures and their characterization, *The Journal of Fuzzy Mathematics* **3**:373-381,(1995)

[9] Fodor, J. and Roubens, M., Fuzzy preference modelling and multi-criteria decision support. Kluwer Academic Publishers, 1994

[10] Gire, F., Well-founded and stable semantics, *J. Logic Programming*, 21(2):95-111,(1994)

[11] Kitaïnik, L, Fuzzy decision procedures with binary relations: towards a unified theory, Kluwer Academic Publ., Boston, 1993

[12] Roy, B., Méthodologie Multicritère d'Aide à la Décision. ECONOMICA, Paris, 1985.

[13] Roy, B., and Bouyssou, D., Aide Multicritère à la Décision: Méthodes et cas. ECONOMICA, Paris, 1993

[14] Van Emden, M.H. and Kowalski, R.A., The semantics of predicate logic as a programming language, *JACM* 23(4):733-742, (1976)