Default Knowledge in Logic Programs with Uncertainty

Yann Loyer[1] and Umberto Straccia[2]

[1] Laboratoire PRiSM, Université de Versailles Saint Quentin
45 Avenue des Etats-Unis, 78035 Versailles FRANCE
[2] I.S.T.I. - C.N.R., Via G. Moruzzi,1 I-56124 Pisa (PI) ITALY

Abstract. Many frameworks have been proposed to manage uncertain information in logic programming. Essentially, they differ in the underlying notion of uncertainty and how these uncertainty values, associated to rules and facts, are managed. The goal of this paper is to allow the reasoning with *non-uniform default assumptions*, i.e. with any arbitrary assignment of default values to the atoms. Informally, rather than to rely on the same default certainty value for all atoms, we allow arbitrary assignments to complete information. To this end, we define both epistemologically and computationally the semantics according to any given assumption. For reasons of generality, we present our work in the framework presented in [17] as a unifying umbrella for many of the proposed approaches to the management of uncertainty in logic programming. Our extension is conservative in the following sense: (i) if we restrict our attention to the usual uniform Open World Assumption, then the semantics reduces to the Kripke-Kleene semantics, and (ii) if we restrict our attention to the uniform Closed World Assumption, then our semantics reduces to the well-founded semantics.

1 Introduction

The management of uncertainty is an important issue whenever the real world information to be represented is of imperfect nature and the classical crisp *true, false* approximation is not adequate (which is likely the rule rather than an exception). Numerous frameworks for logic programming with uncertainty have been proposed. Essentially, they differ in the underlying notion of uncertainty (e.g. probability theory [8, 16, 23–25], fuzzy set theory [3, 27, 29], multi-valued logic [7, 12–14, 17, 28], possibilistic logic [4, 32, 33]) and how uncertainty values, associated to rules and facts, are managed.

Recently, Lakshmanan and Shiri [17] have proposed a general framework, called *Parametric Deductive Databases with Uncertainty* (PDDU), that captures and generalizes many of the precedent approaches, permitting various forms of uncertainty to be manipulated in different ways. Informally, in [17], a rule is of the form $A \stackrel{\alpha}{\leftarrow} B_1, ..., B_n$. Computationally, given an assignment I of certainties, taken from a certainty lattice $\mathcal{L}$, to the B_is, the certainty of A is computed by taking the "conjunction" of the certainties $I(B_i)$ and then somehow "propagating" it to the rule head, taking into account the certainty α of the implication. The term 'parametric' in the PDDU framework derives from the fact it is possible to select, individually for each rule, the aggregation operators, i.e. how conjunction, disjunction and rule propagation are interpreted and, thus, admitting a certain amount of flexibility. Our motivation to consider

the PDDU framework comes from its generality. We will just recall that Lakshmanan and Shiri [17] essentially motivate their parametric approach by the need (i) to permit various forms of uncertainty to be manipulated in different ways, and (ii) to allow for the fact that certain manipulations amount to treating different derivations of an atom as a set (see, e.g. [4, 15, 29]) while others amount to treating it as a multiset (e.g.[16]).

The main topic of this study is to extend the PDDU framework, and thus the approaches captured by that framework, with *non-uniform default assumptions*. Informally, rather than to rely on the same default value for all atoms, we allow arbitrary assignments of default certainty values. An assumption defines default knowledge to be used to complete the available knowledge provided by the facts and rules of a program. As shown in [21], the usual semantics of logic programs can be obtained through a unique computation method, but using different *uniform* assumptions, i.e. assumptions that assign the same default truth-value to all the atoms. Well-known are (i) the *Closed World Assumption* (CWA), which asserts that any atom whose truth-value cannot be inferred from the facts and rules is supposed to be false, and (ii) the *Open World Assumption* (OWA), which asserts that every such atom is supposed to be unknown or undefined. In the first case, fall the approaches, which end up with a *stable model* [9, 10], or *well-founded* [31] semantics, like [5, 19, 23, 24], while in the latter case fall the others, including [17] as well. Exceptions are a previous attempt [19] and those works related to Extended and Disjunctive Logic Programs [10], where some non-uniformity is allowed (see, e.g. [32, 33]). In those works, some atoms are allowed to be interpreted according to the OWA, while others are allowed to be interpreted according to the CWA and, thus, roughly the choice of the default is restricted to the value *unknown* and/or *false*. The general case, where *any* assignment may be considered as the default knowledge, is approached in [22], but relying on a weak evaluation of formulas (based on the possible values of unknown atoms), and thus leads to a weaker semantics than the one presented here (that relies on classical formulas evaluation). To the best of our knowledge there is no other work addressing the general case. To illustrate the idea of non-uniform assumptions, consider the following example. Examples motivating the need for combining OWA and CWA can be found in [10, 22, 32, 33] and in [8].

Example 1. A judge is collecting information from two sources, the public prosecutor and the counsel for the defence, in order to decide whether to charge a person named Ted, accused of murder[3]. The judge first collects a set of facts $F = \{\neg\texttt{has\_witness}(\texttt{Ted}), \texttt{friends}(\texttt{John}, \texttt{Ted})\}$. Then he combines them with his own set of rules R (described below) in order to make a decision.

$$
\begin{array}{ll}
\texttt{suspect}(\texttt{X}) & \leftarrow \texttt{motive}(\texttt{X}) \\
\texttt{suspect}(\texttt{X}) & \leftarrow \texttt{has\_witness}(\texttt{X}) \\
\texttt{cleared}(\texttt{X}) & \leftarrow \texttt{alibi}(\texttt{X}, \texttt{Y}) \wedge \neg\texttt{friends}(\texttt{X}, \texttt{Y}) \\
\texttt{cleared}(\texttt{X}) & \leftarrow \texttt{innocent}(\texttt{X}) \wedge \neg\texttt{suspect}(\texttt{X}) \\
\texttt{friends}(\texttt{X}, \texttt{Y}) & \leftarrow \texttt{friends}(\texttt{Y}, \texttt{X}) \\
\texttt{friends}(\texttt{X}, \texttt{Y}) & \leftarrow \texttt{friends}(\texttt{X}, \texttt{Z}) \wedge \texttt{friends}(\texttt{Z}, \texttt{Y}) \\
\texttt{charge}(\texttt{X}) & \leftarrow \texttt{suspect}(\texttt{X}) \\
\texttt{charge}(\texttt{X}) & \leftarrow \neg\texttt{cleared}(\texttt{X})
\end{array}
$$

The question is: What should the value of $\texttt{charge}(\texttt{Ted})$ be? According to that set of rules, Ted should be charged if he has been proved effectively suspect or not cleared.

[3] For ease, we omit any association of uncertainty numbers to the rules and facts.

Using F and R (and intuition), it appears that uniform default assumption are not appropriate in such a case. If the judge relies on the CWA, then he will decide that `Ted` is not cleared and must be charged despite the absence of proof. Relying on the OWA, the atoms `suspect(Ted)`, `cleared(Ted)` and `charged(Ted)` will be unknown, and the judge cannot take a decision (approaches based on OWA are often too weak). Assuming that by default the atoms `motive(Ted)`, `has_witness(Ted)` and `suspect(Ted)` are false, that the atom `innocent(Ted)` is true and that the others are unknown seems to be an appropriate assumption here. Relying on this assumption, the judge will infer that `Ted` is cleared, not suspect and should not be charged. If the set of certainty values is the unit interval $[0, 1]$, then assuming that by default the value of `motive(Ted)`, `has_witness(Ted)` and `suspect(Ted)` is 0.1, that the value of `innocent(Ted)` is 0.9 and that the others are unknown would be a way to use uncertainty values to distinguish the notion of proved innocence from the notion of presumption of innocence.

Motivation for allowing the use of any assignment of truth values as an assumption, apart from proposing a general framework for reasoning with incomplete information, comes mainly from the area of information integration. Information is now usually scattered in many different sources, and then has to be integrated before to be used. But an information source that received knowledge from other sources may want to privilege its own knowledge. To this end, the source should have the possibility to use the "external" information just to complete its own "internal" knowledge, but without 'removing' it. It follows that the source should have the possibility of considering the external knowledge provided by other sources as an assumption over its own universe. Let us consider the following example to illustrate that situation.

Example 2. Consider an insurance company that has information about its customers (data grouped into a set F of facts, and a set R of rules, i.e. a logic program). That information is used to determine, or to re-evaluate, the risk coefficient in $[0, 1]$ of each customer in order to fix the price of the insurance contracts. In presence of incomplete information, the insurance company may use knowledge provided by other sources such as the risk coefficient of a new customer provided by its precedent insurance company. Such knowledge should be seen as "assumed" or "possible" knowledge for completing the knowledge given by the program, but without overwriting it.

We will rely on the central notion of *support*, which is roughly speaking the maximal knowledge provided by an assumption that can be "safely" added to a program. That notion will be used to define new families of models, called *supported models* which are models that can not be completed anymore by the assumption. As a side effect, our approach extends PDDU, and thus the underlying formalisms, by allowing negation, and generalizes to those frameworks the usual semantics of logic programs with negation. Indeed, our extension of the PDDU framework to non-uniform default assumptions is conservative, i.e. if we restrict our attention to the uniform OWA, then our semantics reduces to the Kripke-Kleene semantics [6]. On the other hand, if we restrict our attention to the uniform CWA, then our semantics captures the well-founded semantics [31].

In the following section, we introduce the syntax of PDDU with negation, called normal parametric programs. Section 3 contains the definitions of interpretation and model of a program. Section 4 presents both epistemic and fixpoint characterizations of

the intended semantics of normal parametric programs with respect to any given default assumption. Section 5 presents some concluding remarks.

2 Preliminaries

We start with some basic definitions related to PDDU [17]. Consider an arbitrary first order language that contains infinitely many variable symbols, finitely many constants, and predicate symbols, but no function symbols. The predicate symbol $\pi(A)$ of an atom A given by $A = p(X_1, \ldots, X_n)$ is defined by $\pi(A) = p$. Let $\mathcal{L} = \langle \mathcal{T}, \preceq, \otimes, \oplus \rangle$ be a *certainty lattice* (a complete lattice) and $\mathcal{B}(\mathcal{T})$ the set of finite multisets over $\mathcal{T}$. A multiset is indicated with $\{\!\{\cdot\}\!\}$. With $\bot$ and $\top$ we denote the least and greatest element in $\mathcal{T}$, respectively. Essentially, as we will see in the next section, an atom will be mapped into a certainty value in $\mathcal{T}$. Typical certainty lattices are: given a set of real values $\mathcal{T}$, define $\mathcal{L}_{\mathcal{T}} = \langle \mathcal{T}, \preceq, \otimes, \oplus \rangle$, $\alpha \preceq \beta$ iff $\alpha \leq \beta$, $\alpha \otimes \beta = \min(\alpha, \beta)$, $\alpha \oplus \beta = \max(\alpha, \beta)$, $\bot = \inf(\mathcal{T})$ and $\top = \sup(\mathcal{T})$, respectively. Then $\mathcal{L}_{\{0,1\}}$ corresponds to the classical truth-space, while $\mathcal{L}_{[0,1]}$, which relies on the unit interval, is quite frequently used as certainty lattice. While the language does not contain function symbols, it contains symbols for families of propagation ($\mathcal{F}_p$), conjunction ($\mathcal{F}_c$) and disjunction functions ($\mathcal{F}_d$), called *combination functions*. The intention is that a conjunction function (e.g. $\otimes$) determines the certainty of a conjunction in the body of a rule, the propagation function (e.g. $\otimes$) determines how to "propagate" the certainty resulting from the evaluation of the body of a rule to the head, by taking into account the certainty associated to the rule, while the disjunction function (e.g. $\oplus$) dictates how to combine the certainties in case an atom appears in the heads of several rules. Combination functions have to satisfy some restrictions to reflect their use. Formally, a *propagation function* is a mapping from $\mathcal{T} \times \mathcal{T}$ to $\mathcal{T}$ and a *conjunction* or *disjunction* function is a mapping from $\mathcal{B}(\mathcal{T})$ to $\mathcal{T}$. Each combination function is monotonic and continuous w.r.t. (with respect to) each one of its arguments. Conjunction and disjunction functions are commutative and associative. Additionally, each kind of function must verify some of the following properties (for simplicity, we formulate the properties treating any function as a binary function on $\mathcal{T}$):

(i) bounded-above: $f(\alpha_1, \alpha_2) \preceq \alpha_i$, for $i = 1, 2, \forall \alpha_1, \alpha_2 \in \mathcal{T}$;
(ii) bounded-below: $\alpha_i \preceq f(\alpha_1, \alpha_2)$, for $i = 1, 2, \forall \alpha_1, \alpha_2 \in \mathcal{T}$;
(iii) $f(\{\!\{\alpha\}\!\}) = \alpha, \forall \alpha \in \mathcal{T}$;
(iv) $f(\emptyset) = \bot$;
(v) $f(\emptyset) = \top$; and
(vi) $f(\alpha, \top) = \alpha, \forall \alpha \in \mathcal{T}$.

The following should be satisfied:

- a conjunction function in $\mathcal{F}_c$ should satisfy properties (i), (iii), (v) and (vi);
- a propagation function in $\mathcal{F}_p$ should satisfy properties (i) and (vi); while
- a disjunction function in $\mathcal{F}_d$ should satisfy properties (ii), (iii) and (iv).

Note that, condition (iii) accounts for the evaluation a one-element conjunction or disjunction; condition (v) deals with a conjunction of an empty body, i.e. a fact, and, thus, the body is evaluated to $\top$, while condition (iv) specifies that a disjunction of an empty

set of literals is evaluated to $\bot$ (for instance, in case an atom is head of no rule). We also assume that there is a function from $\mathcal{T}$ to $\mathcal{T}$, called *negation function*, denoted $\neg$, that is anti-monotone w.r.t. $\preceq$ and satisfies $\neg\neg\alpha = \alpha, \forall\alpha \in \mathcal{T}$. E.g., in $\mathcal{L}_{[0,1]}$, $\neg\alpha = 1 - \alpha$ is quite typical. Finally, a literal is an atomic formula or its negation.

In the following definition of normal parametric program, we extend [17] by allowing literals to appear in the body of a rule. But, as we will clarify later on, in this special case, the role of a literal $\neg A$ depends on the default assumption made on the atom A. If it is the OWA, i.e. by default A's value is unknown, then $\neg A$ behaves classically, if it is the CWA, i.e. by default A's value is false, then $\neg A$ behaves like $not\ A$ (negation as failure). Note that A's default value may well be any other value, e.g. 0.7 in $\mathcal{L}_{[0,1]}$.

Definition 1 (Normal parametric program). *A* normal parametric program P *(np-program) is a 5-tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{C}, \mathcal{P}, \mathcal{D} \rangle$*, whose components are defined as follows:*

1. $\mathcal{L} = \langle \mathcal{T}, \preceq, \otimes, \oplus \rangle$ *is a complete lattice, where* $\mathcal{T}$ *is a set of certainties partially ordered by* $\preceq$*,* $\otimes$ *is the meet operator and* $\oplus$ *the join operator;*
2. $\mathcal{R}$ *is a finite set of* normal parametric rules *(np-rules), each of which is a statement of the form:* $r : A \overset{\alpha_r}{\leftarrow} L_1, ..., L_n$ *where* A *is an atom,* $L_1, ..., L_n$ *are literals or values in* $\mathcal{T}$ *and* $\alpha_r \in \mathcal{T} \setminus \{\bot\}$ *is the certainty of the rule;*
3. $\mathcal{C}$ *maps each np-rule into a conjunction function in* $\mathcal{F}_c$*;*
4. $\mathcal{P}$ *maps each np-rule into a propagation function in* $\mathcal{F}_p$*;*
5. $\mathcal{D}$ *maps each predicate symbol in P into a disjunction function in* $\mathcal{F}_d$*.*

For ease of presentation, we write $r : A \overset{\alpha_r}{\leftarrow} L_1, ..., L_n; \langle f_d, f_p, f_c \rangle$ to represent an np-rule in which $f_d \in \mathcal{F}_d$ is the disjunction function associated with $\pi(A)$ and, $f_c \in \mathcal{F}_c$ and $f_p \in \mathcal{F}_p$ are the conjunction and propagation functions associated with r, respectively. Note that under the above representation, rules involving the same predicate symbol in the head, should have the same disjunction function associated. The *Herbrand base* $\mathcal{B}_P$ of an np-program P is the set of all instantiated program atoms and, P^* is the *Herbrand instantiation of* P, i.e. the set of all ground instantiations of the rules in P (P^* is finite). Note that any Datalog program with negation can be seen as an np-program.

We recall from [17] the following examples, which both might help informally the reader to get confidence with the formalism and show how PDDU may capture different approaches to the management of uncertainty in logic programming. Consider the following generic program with the four rules r_i, $(r_1\colon A \overset{\alpha_1}{\leftarrow} B), (r_2\colon A \overset{\alpha_2}{\leftarrow} C), (r_3\colon B \overset{\alpha_3}{\leftarrow}), (r_4\colon C \overset{\alpha_4}{\leftarrow})$ where A, B, C are ground atoms, where α_i is the certainty of the rule r_i.

Example 3 (Classical case). Consider $\mathcal{L}_{\{0,1\}}$ and $\alpha_i = 1$, for $1 \leq i \leq 4$. Suppose the propagation and conjunction functions associated with r_i are both $\min$, and the disjunction function associated with each atom is $\max$. Then, P is a program in the standard logic programming framework.

Example 4 ([4]). Consider $\mathcal{L}_{[0,1]}$. Suppose $\alpha_1 = 0.8, \alpha_2 = \alpha_3 = 0.7$, and $\alpha_4 = 0.8$ are possibility/necessity degrees associated with the implications. Suppose the conjunction, propagation, and disjunction functions are the same as in the above example. Then P is a program in the framework proposed by Dubois et al. [4], which is founded on Zadeh's possibility theory [35]. In a fixpoint evaluation of P, the possibility/necessity degrees derived for A, B, C are 0.7, 0.7, 0.8, respectively.

Example 5 ([29]). Consider $\mathcal{L}_{[0,1]}$ and suppose α_i as defined in Example 4. Suppose the conjunction and disjunction functions are as before, but the propagation function is multiplication $(\cdot)$. Then P is a program in van Emden's framework [29], which is mathematically founded on the theory of fuzzy sets proposed by Zadeh [34]. In a fixpoint evaluation of P, the values derived for A, B, C are 0.56, 0.7, 0.8, respectively.

Example 6 (MYCIN [2]). Consider $\mathcal{L}_{[0,1]}$, and suppose α_i's are probabilities defined as in the previous example. Suppose $(\cdot)$ is the propagation and conjunction function associated with every rule in P, and $f_d(\alpha, \beta) = \alpha + \beta - \alpha \cdot \beta$ is the disjunction function associated with every atom in P (algebraic sum). Viewing an atom as an event, f_d returns the probability of the occurrence, of any one of two independent events, in the probabilistic sense. Note that f_d is the disjunction function used in MYCIN [2]. Let us consider a fixpoint evaluation of P. In the first step, we derive B and C with probabilities 0.7 and 0.8, respectively. In step 2, applying r_1 and r_2, we obtain two derivations of A, the probability of each of which is 0.56. The probability of A is then defined as $f_d(0.56, 0.56) = 0.8064$. Note that, in this example, collecting derivations as a multiset is crucial; if the derivations were collected as a set, the certainty obtained for A would be 0.56, which is incorrect.

Finally, let us extend the introductory Example 1 to an np-program formulation.

Example 7. Consider the certainty lattice is $\mathcal{L}_{[0;1]}$. Consider $f_d(\alpha, \beta) = \alpha + \beta - \alpha \cdot \beta$, $f_c(\alpha, \beta) = \alpha \cdot \beta$, $f_p = f_c$ and $\neg\alpha = 1 - \alpha$. Let P be the set of rules in Example 1, rewritten as np-program by including uncertainty values, and extended as follows.

$$
\begin{array}{ll}
\texttt{suspect(X)} & \overset{0.6}{\leftarrow} \texttt{motive(X)}\ \langle f_d, \otimes, \otimes\rangle \\
\texttt{suspect(X)} & \overset{0.8}{\leftarrow} \texttt{has\_witness(X)}\ \langle f_d, \otimes, \otimes\rangle \\
\texttt{cleared(X)} & \overset{1}{\leftarrow} \texttt{alibi(X,Y)} \wedge \neg\texttt{friends(X,Y)}\ \langle f_d, f_p, \otimes\rangle \\
\texttt{cleared(X)} & \overset{1}{\leftarrow} \texttt{innocent(X)} \wedge \neg\texttt{suspect(X)}\ \langle f_d, f_p, \otimes\rangle \\
\texttt{friends(X,Y)} & \overset{1}{\leftarrow} \texttt{friends(Y,X)}\ \langle \oplus, f_p, \otimes\rangle \\
\texttt{friends(X,Y)} & \overset{0.7}{\leftarrow} \texttt{friends(X,Z)} \wedge \texttt{friends(Z,Y)}\ \langle \oplus, f_p, f_c\rangle \\
\texttt{charge(X)} & \overset{1}{\leftarrow} \texttt{suspect(X)}\ \langle \oplus, f_p, \otimes\rangle \\
\texttt{charge(X)} & \overset{1}{\leftarrow} \neg\texttt{cleared(X)}\ \langle \oplus, f_p, \otimes\rangle \\
\texttt{motive(Ted)} & \overset{1}{\leftarrow} 1\ \langle \oplus, f_p, \otimes\rangle \\
\texttt{alibi(Ted, John)} & \overset{1}{\leftarrow} 1\ \langle \oplus, f_p, \otimes\rangle \\
\texttt{friends(Ted, John)} & \overset{1}{\leftarrow} 0.8\ \langle \oplus, f_p, \otimes\rangle
\end{array}
$$

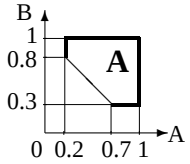
Note that e.g. for predicate `suspect`, the disjunction function is f_d to take into account that if there are different ways for inferring that someone is suspect, then we would like to increase (summing up) our suspicion and not just to choose the maximal value.

3 Interpretations of programs

An *interpretation* I of an np-program P is a function that assigns to all atoms of the Herbrand base of P a value in $\mathcal{T}$. The semantics of a program P is usually an interpretation chosen in the set of models of P. In Datalog programs, as well as in PDDU,

that chosen model is generally the least model of P w.r.t. $\preceq$.[4] Due to the non-monotone negation operator on lattices, some logic programs do not have a unique minimal model anymore, as shown in the following example.

Example 8. Consider the certainty lattice $\mathcal{L}_{[0,1]}$ and the program $P = \{(A \leftarrow \neg B), (B \leftarrow \neg A), (A \leftarrow 0.2), (B \leftarrow 0.3)\}$. Informally, an interpretation I is a model of the program if it satisfies every rule, while I satisfies a rule $X \leftarrow Y$ if $I(X) \succeq I(Y)$. So, this program has an infinite number of models I_x^y, where $0.2 \preceq x \preceq 1$, $0.3 \preceq y \preceq 1$, $y \geq 1 - x$, $I_x^y(A) = x$ and $I_x^y(B) = y$. (those in the A area below). There are also an infinite number of minimal models (those on the thin diagonal line). The minimal models I_x^y are such that $y = 1 - x$.



Concerning the previous example we may note that the certainty of A in any minimal models of P belongs to the interval $[0.2, 0.7]$, while for B the interval is $[0.3, 0.8]$. An obvious question is: what should be the answer to a query A to the program proposed in Example 8? There are at least two answers:

1. the certainty of A is *undefined*, as there is no unique minimal model. This is a conservative approach, which in case of ambiguity prefers to leave A unspecified;
2. the certainty of A is in $[0.2, 0.7]$, which means that even if there is no unique value for A, in all minimal models the certainty of A is in $[0.2, 0.7]$. In this approach we still try to provide some information. Of course, some care should be used. Indeed from $I(A) \in [0.2, 0.7]$ and $I(B) \in [0.3, 0.8]$ we should not conclude that $I(A) = 0.2$ and $I(B) = 0.3$ is a model of the program.

Applying a usual approach, like the well-founded semantics [31] or the Kripke-Kleene semantics [6], would lead us to choose the conservative solution 1. This was also the approach taken in [19]. Such a semantics seems to be too weak, in the sense that it loses some knowledge (e.g. the value of A should be at least 0.2). In this paper we rely on the more informative solution 2.

There is a well-known solution to it that consists in relying on the induced *interval bilattice* (these bilattices are obtained in a standard way as product of a lattice by itself –see, for instance [5, 11]). Indeed, we propose to rely on $\mathcal{T} \times \mathcal{T}$. Any element of $\mathcal{T} \times \mathcal{T}$ is denoted by $[a; b]$ and interpreted as an interval on $\mathcal{T}$, i.e. $[a; b]$ *is interpreted as the set of elements* $x \in \mathcal{T}$ *such that* $a \preceq x \preceq b$. For instance, turning back to Example 8 above, in the intended model of P, the certainty of A is "approximated" with $[0.2; 0.7]$, i.e. the certainty of A lies in between 0.2 and 0.7 (similarly for B). Formally, given a complete lattice $\mathcal{L} = \langle \mathcal{T}, \preceq, \otimes, \oplus \rangle$, we construct a *bilattice* over $\mathcal{T} \times \mathcal{T}$. We recall that a bilattice is a triple $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$, where $\mathcal{B}$ is a nonempty set and $\preceq_t$, $\preceq_k$ are both partial orderings giving to $\mathcal{B}$ the structure of a lattice with a top and a bottom [11]. In our setting, we consider $\mathcal{B} = \mathcal{T} \times \mathcal{T}$ with the two following orderings:

[4] $\preceq$ is extended to the set of interpretations as follows: $I \preceq J$ iff $I(A) \preceq J(A)$ for all atoms A.

– the *truth ordering* $\preceq_t$, where $[a;b] \preceq_t [a';b']$ iff $a \preceq a'$ and $b \preceq b'$;
– the *knowledge ordering* $\preceq_k$, where $[a;b] \preceq_k [a';b']$ iff $a \preceq a'$ and $b' \preceq b$.

The intuition of those orders is that truth increases if the interval contains greater values (e.g. $[0.1;0.4] \preceq_t [0.2;0.5]$), whereas the knowledge increases when the interval (i.e. in our case the approximation of a certainty value) becomes more precise (e.g. $[0.1;0.4] \preceq_k [0.2;0.3]$, i.e. we have more knowledge)[5]. The least and greatest elements of $\mathcal{T} \times \mathcal{T}$ are respectively (i) $\mathtt{f} = [\bot;\bot]$ (false) and $\mathtt{t} = [\top;\top]$ (true), w.r.t. $\preceq_t$; and (ii) $\bot = [\bot;\top]$ (unknown – the less precise interval, i.e. the atom's certainty value is unknown) and $\top = [\top;\bot]$ (inconsistent – the empty interval) w.r.t. $\preceq_k$. The meet, join and negation on $\mathcal{T} \times \mathcal{T}$ w.r.t. both orderings are defined by extending the meet, join and negation from $\mathcal{T}$ to $\mathcal{T} \times \mathcal{T}$ in the natural way: let $[a;b], [a';b'] \in \mathcal{T} \times \mathcal{T}$, then

– $[a;b] \otimes^t [a';b'] = [a \otimes a', b \otimes b']$ and $[a;b] \oplus^t [a';b'] = [a \oplus a', b \oplus b']$;
– $[a;b] \otimes^k [a';b'] = [a \otimes a', b \oplus b']$ and $[a;b] \oplus^k [a';b'] = [a \oplus a', b \otimes b']$;
– $\neg[a;b] = [\neg b; \neg a]$.

$\otimes^t$ and $\oplus^t$ ($\otimes^k$ and $\oplus^k$) denote the meet and join operations on $\mathcal{T} \times \mathcal{T}$ w.r.t. the truth (knowledge) ordering, respectively. For instance, in $\mathcal{L}_{[0,1]}$,

$$\begin{array}{ll} [0.1;0.4] \oplus^t [0.2;0.5] & = [0.2;0.5] \ , \\ [0.1;0.4] \otimes^t [0.2;0.5] & = [0.1;0.4] \ , \\ [0.1;0.4] \oplus^k [0.2;0.5] & = [0.2;0.4] \ , \\ [0.1;0.4] \otimes^k [0.2;0.5] & = [0.1;0.5] \ , \\ \neg[0.1;0.4] & = [0.6;0.9] \ . \end{array}$$

Finally, we extend in a similar way the combination functions from $\mathcal{T}$ to $\mathcal{T} \times \mathcal{T}$. Let f_c (resp. f_p and f_d) be a conjunction (resp. propagation and disjunction) function over $\mathcal{T}$ and $[a;b], [a';b'] \in \mathcal{T} \times \mathcal{T}$, then

– $f_c([a;b],[a';b']) = [f_c(a,a'), f_c(b,b')]$;
– $f_p([a;b],[a';b']) = [f_p(a,a'), f_p(b,b')]$;
– $f_d([a;b],[a';b']) = [f_d(a,a'), f_d(b,b')]$.

It is easy to verify that the combination functions above preserve the original properties of combination functions and that the negation function is monotone w.r.t. $\preceq_k$.

Theorem 1. *Consider $\mathcal{T} \times \mathcal{T}$ with the orderings $\preceq_t$ and $\preceq_k$. Then*

1. *$\otimes^t, \oplus^t, \otimes^k, \oplus^k$ and the extensions of combination functions are continuous (and, thus, monotonic) w.r.t. $\preceq_t$ and $\preceq_k$;*
2. *any extended negation function is monotonic w.r.t. $\preceq_k$;*
3. *if the negation function satisfies the de Morgan laws, i.e. $\forall a, b \in \mathcal{T}. \neg(a \oplus b) = \neg a \otimes \neg b$ then the extended negation function is continuous w.r.t. $\preceq_k$.*

We extend interpretations based on $\mathcal{T}$ to $\mathcal{T} \times \mathcal{T}$ as follows.

[5] An alternative to the interval bilattice is used in [5, 11], where a pair $[a;b]$ is interpreted as degree of doubt and degree of belief, or, similarly, as degree of falsity and degree of truth, respectively. Of course definitions of orders and extensions of functions would then be different, but the whole theory presented in the paper would still hold.

Definition 2 (Approximate interpretation). *Let P be an np-program. An* approximate interpretation *of P is a total function I from the Herbrand base $\mathcal{B}_P$ to the set $\mathcal{T} \times \mathcal{T}$. The set of all the approximate interpretations of P is denoted $\mathcal{C}_P$.*

Intuitively, assigning the logical value $[a; b]$ to an atom A means that the exact certainty value of A lies in between a and b w.r.t. $\preceq$. $I(A) = [a; b]$ can also be understood as the specification of a *lower bound and upper bound constraint* on the admissible certainty values of A. Our goal will be to determine for each atom of the Herbrand base of P the most precise interval that can be inferred.

The two orderings on $\mathcal{T} \times \mathcal{T}$ can be extended to the set of approximate interpretations $\mathcal{C}_P$ in a usual way: let I_1 and I_2 be in $\mathcal{C}_P$, $I_1 \preceq_t I_2$ iff $I_1(A) \preceq_t I_2(A)$, for all ground atoms A; and $I_1 \preceq_k I_2$ iff $I_1(A) \preceq_k I_2(A)$, for all ground atoms A. Under these two orderings $\mathcal{C}_P$ becomes a complete bilattice. The meet and join operations over $\mathcal{T} \times \mathcal{T}$ for both orderings are extended to $\mathcal{C}_P$ in the usual way (e.g. for any atom A, $(I \oplus^k J)(A) = I(A) \oplus^k J(A)$). Negation is extended similarly, for any atom A, $\neg I(A) = I(\neg A)$, and approximate interpretations are extended to $\mathcal{T}$, for any $\alpha \in \mathcal{T}$, $I(\alpha) = [\alpha; \alpha]$. With $\mathtt{I_f}$ and $\mathtt{I_t}$ we will denote the bottom and top approximate interpretation under $\preceq_t$ (they map any atom into $\mathtt{f}$ and $\mathtt{t}$, respectively). With $\mathtt{I}_\bot$ and $\mathtt{I}_\top$ we will denote the bottom and top approximate interpretation under $\preceq_k$ (they map any atom into $\bot$ and $\top$, respectively).

Definition 3 (Model of a logic program). *Let P be an np-program and let I be an approximate interpretation of P.*

1. I satisfies *a ground np-rule* $r\colon A \overset{\alpha_r}{\leftarrow} L_1, ..., L_n; \langle f_d, f_p, f_c\rangle$ *in P, denoted* $\models_I r$*, iff* $I(A) \succeq_t f_p([\alpha_r; \alpha_r], f_c(\{I(L_1), \ldots, I(L_n)\}))$*;*
2. I *is a* model *of P, or I* satisfies *P, denoted* $\models_I P$*, iff for all atoms* $A \in \mathcal{B}_P$*,* $I(A) \succeq_t f_d(X)$ *where f_d is the disjunction function associated with $\pi(A)$ and*

$$X = \{f_p([\alpha_r; \alpha_r], f_c(\{I(L_1), \ldots, I(L_n)\}))\colon A \overset{\alpha_r}{\leftarrow} L_1, ..., L_n; \langle f_d, f_p, f_c\rangle \in P^*\}\,.$$

4 Assumption-based semantics

In this section we define the semantics of an np-program w.r.t. a given assumption.

Definition 4 (assumption). *An* assumption H *is an approximate interpretation.*

Informally, we propose to complete our knowledge using an approximate interpretation over the Herbrand base. The term assumption is used in order to stress the fact that it represents "default" knowledge and not the usual "sure" knowledge.

At first, we give the definition of the immediate consequence operator over approximate interpretations, which does not take into account assumptions.

Definition 5. *Let P and I be an np-program and an approximate interpretation, respectively. The* immediate consequence operator T_P *is defined as follows: for every atom A, $T_P(I)(A) = f_d(X)$, where f_d is the disjunction function associated with $\pi(A)$ and*

$$X = \{f_p([\alpha_r; \alpha_r], f_c(\{I(L_1), \ldots, I(L_n)\}))\colon A \overset{\alpha_r}{\leftarrow} L_1, ..., L_n; \langle f_d, f_p, f_c\rangle \in P^*\}\,.$$

Note that by property (iv) of combination functions satisfied by all disjunction functions, it follows that if an atom A does not appear as the head of a rule, then $T_P(I)(A) = \mathtt{f}$. It can be shown that

Theorem 2. *For any np-program P, T_P is monotonic and, if the de Morgan laws hold, continuous w.r.t. $\preceq_k$.*

It is worth noting that the least fixpoint of T_P with respect to $\preceq_k$ (which exists as T_P is monotonic w.r.t. $\preceq_k$) corresponds to an extension of the classical Kripke-Kleene semantics [6] of Datalog programs with negation to np-programs. Indeed, if we restrict our attention to Datalog with negation, then we have to deal with four values $[0;0], [1;1], [0;1]$ and $[1;0]$ that correspond to the truth values *false, true, unknown* and *inconsistent*, respectively. In such a setting, our bilattice coincides with Belnap's bilattice $\mathcal{FOUR}$ [1] and for any Datalog program with negation P, the least fixpoint of T_P w.r.t. $\preceq_k$ is a model of P that coincides with the Kripke-Kleene semantics of P.

Example 9. Consider $P = \{(A \leftarrow B, C), (C \leftarrow C, D), (B \leftarrow 0.7), (D \leftarrow 0.9)\}$[6], and $\mathcal{L}_{[0,1]}$ as certainty lattice. The maximal knowledge I that can be inferred from that program (without any extra knowledge such as assumptions) is that the (minimal) values of B and D are exactly 0.7 and 0.9 respectively, while the values of A and C are at most 0.7 and 0.9 respectively, i.e. $I(A) = [0; 0.7]$, $I(B) = [0.7; 0.7]$, $I(C) = [0; 0.9]$ and $I(D) = [0.9; 0.9]$. Using an assumption H asserting that the value of C is by default at least 0.6, i.e. $H(C) = [0.6; 1]$, then we should come up with a more precise characterisation of A and C by stating that A is in $[0.6, 0.7]$ and C is in $[0.6, 0.9]$.

The formalisation of the above concept relies on the notion of safe approximation and support. Intuitively, the notion of *support* provided by a hypothesis H to an np-program P and an interpretation I, denoted $s_P^H(I)$, can be explained as follows. We regard I as what we already know about an intended model of P. On the basis of both the current knowledge I and the information expressed by the rules of P, we want to complete our available knowledge I, by using the hypothesis H. We regard the hypothesis as an additional source of information to be used to complete I. The support of I, $s_P^H(I)$, indeed determines in a principled way the amount of information provided by the hypothesis H that can be joined to I. The concept of support has been extensively discussed in [20] in relation to an alternative, epistemic based, characterization of the well-founded semantics over bilattices. But, in [20], the 'hypothesis' is *fixed* to the approximate interpretation $H = \mathtt{I_f}$. That is, by default the certainty of any atom is $\mathtt{f}$. We base our work on [20], rather than try to extend the well-know Gelfond-Lifschitz transformation [9, 10], for two main reasons. First the Gelfond-Lifschitz transformation provides only a computational definition of the semantics whereas our approach provides also an epistemic definition, and thus lights the role of the assumption. Second, the limits of the Gelfond-Lifschitz transformation for reasoning with general assumptions have been shown in [18], where that transformation has been extended for reasoning with any given assumption over Belnap's bilattice $\mathcal{FOUR}$ [1]. It is shown that properties of extremal fixpoints of monotone operators over lattices (and bilattices), on the one hand, allow the use of any such assumption instead of the CWA, but, on the other hand,

[6] For ease of presentation, we omit combination functions.

restrict the set of default values to the set of extremal values only and thus do not allow the use of any assumption over a lattice *different* from $\mathcal{FOUR}$.

Formally, at first, we extend T_P to take an assumption into account as well.

Definition 6. *The* immediate consequence operator T_P^H *is defined as* T_P*, except that rather than mapping into* f *all atoms that do not appear as the head of any rule in* P*,* T_P^H *maps every such atom* A *into its default value* $H(A)$*.*

Definition 7 (safe approximation). *Let* P, I *and* H *be an np-program, an approximate interpretation and an assumption, respectively. An approximate interpretation* J *is* safe *w.r.t.* P, H *and* I *iff both* $J \preceq_k H$ *and* $J \preceq_k T_P^H(I \oplus^k J)$ *hold.*

In the above definition, the first condition dictates that a safe interpretation J must not be more informative than a given assumption H. If $J = H$, then every ground atom assumes its default value. But, given I and P, not necessarily the value of every atom may be given by its default (e.g., for some atoms we may infer something else) and, thus, we have to consider some weaker assumption $J \preceq_k H$. The second item dictates that a safe interpretation must be *cumulative*, i.e. that the accumulated amount of information provided by the assumption should be preserved and lead to more precise approximations of an intended model of P.

Example 10. Consider Example 9 and the following table of approximate interpretations I, J_1, J_2, J and assumption H.

	A	B	C	D
I	$[0; 0.7]$	$[0.7; 0.7]$	$[0; 0.9]$	$[0.9; 0.9]$
H	$[0.4; 0.5]$	$[0; 1]$	$[0.6; 1]$	$[0; 0]$
J_1	$[0.3; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$
J_2	$[0.4; 0.8]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$
J	$[0.4; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$

Then both J_1 and J_2 are safe w.r.t. P, H and I. It is easy to see that the maximal safe approximate interpretation w.r.t. the knowledge order is $J = J_1 \oplus^k J_2$.

Given an assumption H assigning a default value to any atom of the Herbrand base, we can not expect it to be safe, and thus we have to relax the assumption and consider a safe interpretation w.r.t. H, i.e. weaker than H. Among all possible safe approximate interpretations w.r.t. H, we will privilege the maximal one under $\preceq_k$ as our goal is to use it to complete the knowledge provided by P and I.

Definition 8 (support). *Let* P *be an np-program, let* I *be an approximate interpretation and consider an assumption* H*. The* support*, denoted* $s_P^H(I)$*, provided by* H *to* P *and* I *is the maximal safe interpretation w.r.t.* P, H *and* I *under* $\preceq_k$*.*

It can be shown that if J_1 and J_2 are two safe interpretations w.r.t. P, H and I, then $J_1 \oplus^k J_2$ is safe w.r.t. P, H and I. It follows that the support is a well-defined concept and that we have $s_P^H(I) = \bigoplus^k \{J : J \text{ is safe w.r.t. } P, H \text{ and } I\}$.

The following theorem provides an algorithm for computing the support.

Theorem 3. *Let* P *be an np-program.* $s_P^H(I)$ *can be computed as the iterated fixpoint of* $F_{P,I}^H$ *beginning the computation with* H*, where* $F_{P,I}^H(J) = H \otimes^k T_P^H(I \oplus^k J)$*.*

From Theorems 1 and 2, it can be shown that $F^H_{P,I}$ is monotone and, if the de Morgan laws hold, continuous w.r.t. $\preceq_k$. It follows that the iteration of the function $F^H_{P,I}$ starting from H decreases w.r.t. $\preceq_k$ and reaches a fixed point.

Example 11 (Example 10 cont.). Below, the computation of the support is shown.

	A	B	C	D
I	$[0; 0.7]$	$[0.7; 0.7]$	$[0; 0.9]$	$[0.9; 0.9]$
$J_0 = H$	$[0.4; 0.5]$	$[0; 1]$	$[0.6; 1]$	$[0; 0]$
$J_1 = J_2 = s^H_P(I)$	$[0.4; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$

Intuitively, the support of an assumption represents the maximal knowledge that can be considered "compatible" with the program and that can be used "safely" to *complete* the knowledge inferred from the program. Given an assumption H, this intuition leads to the definition of a family of models, those models that already integrate their own support provided by H, i.e. the models of P "supported" by H.

Definition 9 (H-supported models). *Let P be an np-program and let H be an assumption. A model I of P is* supported *by H if $s^H_P(I) \preceq_k I$. The* semantics *of P supported by H is the least model under $\preceq_k$ of P supported by H, denoted A^H_P.*

Example 12 (Example 11 cont.). Below, the semantics, A^H_P, of P supported by H is shown. Note that $s^H_P(A^H_P) \preceq_k H$ and $s^H_P(A^H_P) \preceq_k A^H_P$, i.e. A^H_P contains all the default information that can be used to complete the knowledge provided by the program.

	A	B	C	D
H	$[0.4; 0.5]$	$[0; 1]$	$[0.6; 1]$	$[0; 0]$
$s^H_P(A^H_P)$	$[0.4; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$
A^H_P	$[0.6; 0.7]$	$[0.7; 0.7]$	$[0.6; 0.9]$	$[0.9; 0.9]$

Note that there are two special cases of assumptions that are closely related to the interpretation of negation : given an atom A, then if $H(A) = \mathsf{f}$ then $\neg A$ is interpreted as negation-as-failure $not\ A$, whereas if $H(A) = \bot$ then we revert to the classical interpretation of negation. These two particular cases may be seen as strictly related to Extended and Disjunctive Logic Programs [10], in which you may consider to add a rule of the form $\neg A \leftarrow not\ A$ or not, respectively.

Supported models have also a fixed point characterization. Given an approximate interpretation I, an assumption H and an np-program P, there are two ways of inferring information from an np-program P: (i) using T^H_P; and (ii) using s^H_P. We propose to combine them in order to infer as much knowledge as possible: that is (i) first we compute $s^H_P(I)$, i.e. the most precise approximation of the assumption H that can be used safely; (ii) we add that approximate interpretation to I (that represents the available knowledge) and obtain $I \oplus^k s^H_P(I)$; and finally (iii) we activate the rules of the program by applying the immediate consequence operator, i.e. we compute $T^H_P(I \oplus^k s^H_P(I))$.

Definition 10. *Let P, H and I be an np-program, an assumption and an approximate interpretation, respectively. The* assumption-based immediate consequence operator, *denoted Γ^H_P, is defined as $\Gamma^H_P(I) = T^H_P(I \oplus^k s^H_P(I))$.*

From the monotonicity (continuity) of T^H_P and s^H_P over the complete lattice $\mathcal{C}_P$, w.r.t. $\preceq_k$, by the well-known Knaster-Tarski theorem, it follows that

Theorem 4. *Let P be an np-program and let H be an assumption. Then Γ_P^H is monotone and, if the de Morgan laws hold, continuous w.r.t. the knowledge order $\preceq_k$. Therefore, Γ_P^H has a least fixpoint w.r.t. the knowledge order $\preceq_k$.*

Γ_P^H provides a fixed point characterization of supported models and, thus, A_P^H can be computed as the iterated fixpoint of Γ_P^H starting with $\mathtt{I}_\perp$.

Theorem 5. *Let P be an np-program and let H be an assumption. An interpretation I is a model of P supported by H iff I is a fixpoint of Γ_P^H.*

Example 13 (Example 11 cont.). Below, the computation of the semantics A_P^H is shown.

	A	B	C	D
H	$[0.4; 0.5]$	$[0; 1]$	$[0.6; 1]$	$[0; 0]$
$I_0 = \mathtt{I}_\perp$	$[0; 1]$	$[0; 1]$	$[0; 1]$	$[0; 1]$
$s_P^H(I_0)$	$[0; 1]$	$[0; 1]$	$[0; 1]$	$[0; 0.9]$
I_1	$[0; 1]$	$[0.7; 0.7]$	$[0; 0.9]$	$[0.9; 0.9]$
$s_P^H(I_1)$	$[0.4; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$
I_2	$[0.6; 0.7]$	$[0.7; 0.7]$	$[0.6, 0.9]$	$[0.9; 0.9]$
$s_P^H(I_2)$	$[0.4; 0.7]$	$[0; 1]$	$[0.6; 1]$	$[0; 0.9]$
$I_3 = I_2 = A_P^H$	$[0.6; 0.7]$	$[0.7; 0.7]$	$[0.6; 0.9]$	$[0.9; 0.9]$

Example 14. Consider the program P given in Example 7. Let H be the assumption that assigns to the atoms `motive(Ted)`, `has_witness(Ted)` and `suspect(Ted)` the value $[0; 0]$, to the atom `innocent(Ted)` the value $[1; 1]$, and to the other atoms the value $[0; 1]$. Then it can be shown that the semantics of P for the three different assumptions, $H_1 = \mathtt{I}_\perp$ (i.e. everything unknown), $H_2 = \mathtt{I_f}$ (i.e. everything false) and H are:

	`suspect(Ted)`	`cleared(Ted)`	`charge(Ted)`
$A_P^{H_1}$	$[0.6; 0.92]$	$[0.2; 0.52]$	$[0.6; 0.92]$
$A_P^{H_2}$	$[0.6; 0.6]$	$[0.2; 0.2]$	$[0.8; 0.8]$
A_P^H	$[0.6; 0.6]$	$[0.52; 0.52]$	$[0.6; 0.6]$

Note that using the assumption that everything is unknown by default gives obviously a less precise semantics than using any other assumption. As anticipated in Example 1, the OWA, i.e. H_1, does not provide precise information, and the CWA, i.e. H_2, decreases the degree of innocence in presence of incomplete information and leads to charge a person despite the absence of evidence for or against the guiltyness of that person, while the assumption H respects the principle of presumption of innocence and leads to expected results.

A first attempt to give a semantics to PDDU with negation under non-uniform assumptions is described in [19]. It is based on an extension of the alternating fixpoint semantics of van Gelder [30], i.e. on a separated management of positive and negative literals. The approach proposed here is based on a more intuitive management of negation where no distinction is made between positive and negative literals. Moreover, our approach is more general: in fact (i) in [19], some knowledge is lost whenever the value of an atom oscillates between two different uncertainty values, while in our approach, an approximation (interval) of the uncertainty value is assigned. For instance, with respect to

Example 8 under the assumption $\mathtt{I}_\perp$, [19] assigns $\perp = [0, 1]$ to both A and B, while our approach assigns $[0.2, 0.7]$ to A and $[0.3, 0.8]$ to B; and (ii) our approach does not impose any restriction on the kind of interpretations that can be used as assumptions (any value, even an interval, can be considered as a default value).

If we restrict our attention to PDDU and consider the assumption $H = \mathtt{I_f}$, then our approach captures the semantics proposed in [17] for PDDU.

Theorem 6. *Let P be a program as defined in [17]. If $H = \mathtt{I_f}$, then the semantics A_P^H assigns exact values to all atoms and coincides with the semantics presented in [17].*

The following theorem states that our approach extends the usual semantics of normal logic programs to the PDDU framework.

Theorem 7. *Let P be a Datalog program with negation. Then we have:*

1. *the semantics of P w.r.t. $H = \mathtt{I_f}$, A_P^H, is the well-founded semantics of P;*
2. *if the assumption H is such that, for all atoms A, if A is not the head of any rule in P^*, then $H(A) = \mathtt{f}$ else $H(A) = \perp$, then the semantics A_P^H of P coincides with the Kripke-Kleene semantics of P.*
3. *a stable model of P [9] is a model of P supported by $H = \mathtt{I_f}$.*

5 Conclusions

PDDU [17] has been proposed as a unifying umbrella for many existing approaches towards the manipulation of uncertainty in deductive databases. We introduced non-uniform assumptions into this context, which allows to assign any default value to the atoms. The well known OWA, CWA and the combination of both are, thus, special cases. A main topic for further research is to generalize the introduced concept within a more fundamental work in which we deal with bilattices, disjunctive programs, assumptions and stable models [9, 10, 26] in place of our current setting.

References

1. N. D. Belnap. How a computer should think. In Gilbert Ryle, editor, *Contemporary aspects of philosophy*, pages 30–56. Oriel Press, Stocksfield, GB, 1977.
2. B.G. Buchanan and E.H. Shortli. A model of inexact reasoning in medicine. *Mathematical Bioscience*, 23:351–379, 1975.
3. T. H. Cao. Annotated fuzzy logic programs. *Fuzzy Sets and Systems*, 113(2):277–298, 2000.
4. D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In *Proc. of ICLP-91*, pages 581–595. MIT Press, 1991.
5. M. Fitting. The family of stable models. *J. of Logic Programming*, 17:197–225, 1993.
6. M. Fitting. A Kripke-Kleene-semantics for general logic programs. *J. of Logic Programming*, 2:295–312, 1985.
7. M. Fitting. Bilattices and the semantics of logic programming. *J. of Logic Programming*, 11:91–116, 1991.
8. N. Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *J. of the American Society for Information Science*, 51(2):95–110, 2000.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP-88*, pages 1070–1080, 1988. MIT Press.

10. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.
11. M. L. Ginsberg. Multi-valued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
12. M. Kifer and Ai Li. On the semantics of rule-based expert systems with uncertainty. In *Proc. of ICDT-88*, LNCS 326, pages 102–117, 1988.
13. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming*, 12:335–367, 1992.
14. L. Lakshmanan. An epistemic foundation for logic programming with uncertainty. In LNCS 880, pages 89–100, 1994.
15. L. Lakshmanan and F. Sadri. Uncertain deductive databases: a hybrid approach. *Information Systems*, 22(8):483–508, 1997.
16. L. Lakshmanan and N. Shiri. Probabilistic deductive databases. In *Int'l Logic Programming Symposium*, pages 254–268, 1994.
17. L. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.
18. Y. Loyer and N. Spyratos and D. Stamate. Integration of Information in Four-valued Logics under Non-Uniform Assumptions. In *Proc. of the 30th IEEE Int. Symp. on Multi-Valued Logics*, IEEE Press, pages 185–191, 2000.
19. Y. Loyer and U. Straccia. Uncertainty and partial non-uniform assumptions in parametric deductive databases. In *Proc. of JELIA-02*, LNCS 2424, pages 271–282,
20. Y. Loyer and U. Straccia. The well-founded semantics of logic programs over bilattices: an alternative characterisation. TR ISTI-2003-TR-05, ISTI-CNR, Pisa, Italy, 2003. Submitted.
21. Y. Loyer, N. Spyratos and D. Stamate. Parametrized Semantics of Logic Programs - a unifying approach. *Theoretical Computer Science*, to appear.
22. Y. Loyer, N. Spyratos and D. Stamate. Hypotheses-Based Semantics of Logic Programs in Multi-Valued Logics. *ACM Transactions on Computational Logic*, to appear.
23. T. Lukasiewicz. Fixpoint characterizations for many-valued disjunctive logic programs with probabilistic semantics. In *Proc. of LPNMR-01*, LNCS 2173, pages 336–350, 2001.
24. R. Ng and V. S. Subrahmanian. Stable model semantics for probabilistic deductive databases. In *Proc. of ISMIS-91*, LNCS 542, pages 163–171, 1991.
25. R. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1993.
26. T. C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In *Proc. of ICLP-90*, pages 459–477. MIT Press, 1990.
27. Ehud Y. Shapiro. Logic programs with uncertainties: A tool for implementing rule-based systems. In *Proc. of IJCAI-83*, pages 529–532, 1983.
28. V.S. Subramanian. On the semantics of quantitative logic programs. In *Proc. of 4th IEEE Symp. on Logic Programming*, pages 173–182. Computer Society Press, 1987.
29. M.H. van Emden. Quantitative deduction and its fixpoint theory. *J. of Logic Programming*, 4(1):37–53, 1986.
30. A. van Gelder. The alternating fixpoint of logic programs with negation. In *Proc. of ACM PODS-89*, pages 1–10, 1989.
31. A. van Gelder, K. A. Ross, and J. S. Schlimpf. The well-founded semantics for general logic programs. *J. of the ACM*, 38(3):620–650, January 1991.
32. G. Wagner. A logical reconstruction of fuzzy inference in databases and logic programs. In *Proc. of IFSA-97*, Prague, 1997.
33. G. Wagner. Negation in fuzzy and possibilistic logic programs. In *Logic programming and Soft Computing*, Research Studies Press, 1998.
34. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.
35. L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, 1965.