Non-Uniform Hypothesis in Deductive Databases with Uncertainty

Yann Loyer and Umberto Straccia

Istituto di Elaborazione della Informazione -C.N.R.
Via G. Moruzzi, 1 I-56124 Pisa (PI) ITALY
Number: 2002-TR-02

Abstract

Different many-valued logic programming frameworks has been proposed to manage uncertain information in deductive databases and logic programming. A feature of these frameworks is that they rely on a predefined assumption or hypothesis, i.e. an interpretation that assigns the same default truth value to all the atoms of a program, e.g. in the open world assumption, by default all atoms have unknown truth value. In this paper we extend these frameworks along three directions by introducing non-uniform hypotheses: (i) we will deal with the non-monotonic mode of negation; (ii) the default truth values of atoms need not necessarily to be all equal each other; and (iii) a hypothesis can be a partial interpretation. We will show that our approach extends the usual ones: if we restrict our attention to classical logic programs and consider total uniform hypotheses, then our semantics reduces to the usual semantics of logic programs.

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

1 Introduction

An important issue to be addressed in applications of logic programming is the management of uncertainty whenever the information to be represented is of imperfect nature (which happens quite often). The problem of uncertainty management in logic programs has attracted the attention of many researchers

E-mail: straccia@iei.pi.cnr.it

and numerous frameworks have been proposed \[1–3,6,8,11–16,18–22,27–31\]. Each of them addresses the management of different kind of uncertainty: (i) probability theory \[8,15,19–22,31\]; (ii) fuzzy set theory \[1,27,29,30\]; (iii) multi-valued logic \[6,12,13,16,18\]; and (iv) possibilistic logic \[3\]. Apart from the different notion of uncertainty they rely on, these frameworks differ in the way in which uncertainty is associated with the facts and rules of a program. With respect to this latter point, these frameworks can be classified into annotation based (AB) and implication based (IB), which we briefly summarize below. In the AB approach, a rule is of the form 

\[ A : f(\beta_1, \ldots, \beta_n) \leftarrow B_1 : \beta_1, \ldots, B_n : \beta_n, \]

which asserts “the certainty of atom \(A\) is at least (or is in) \(f(\beta_1, \ldots, \beta_n)\), whenever the certainty of atom \(B_i\) is at least (or is in) \(\beta_i\), \(1 \leq i \leq n\)”. Here \(f\) is an \(n\)-ary computable function and \(\beta_i\) is either a constant or a variable ranging over an appropriate certainty domain. Examples of AB frameworks include \[12,13,20–22,28\]. In the IB approach, a rule is of the form 

\[ A \leftarrow B_1, \ldots, B_n, \]

which says that the certainty associated with the implication \(B_1 \land \ldots \land B_n \rightarrow A\) is \(\alpha\). Computationally, given an assignment \(v\) of certainties to the \(B_i\)'s, the certainty of \(A\) is computed by taking the “conjunction” of the certainties \(v(B_i)\) and then somehow “propagating” it to the rule head. The truth values are taken from a certainty lattice. Examples of the IB frameworks include \[4,6,14–16,29\] (see \[16\] for a more detailed comparison between the two approaches).

We limit our contribution in this sense to recall the following facts \[16\]: (i) while the way implication is treated in the AB approach is closer to classical logic, the way rules are fired in the IB approach has a definite intuitive appeal and (ii) the AB approach is strictly more expressive than the IB. The downside is that query processing in the AB approach is more complicated, \(e.g.\) the fixpoint operator is not continuous in general, while it is in the IB approaches. From the above points, it is believed that the IB approach is easier to use and is more amenable for efficient implementation. Nonetheless a common feature of both approaches is that they rely on a predefined uniform assumption or uniform hypothesis, \(i.e.\) the assumption made concerning the atoms whose logical values cannot be inferred from the rules. For instance, in the AB approach the Open World Assumption (OWA) is used, which corresponds to the assumption that every such atom has unknown truth value, while in the IB approach the hypothesis is that every such atom has the bottom element of a truth lattice as its default truth value.

While uniform assumptions are widely used, there are cases where non-uniformity is suitable, as the following example highlights. \(^2\)

**Example 1 (A motivating example)** Consider a legal case where a judge

\(^2\) The need of non-uniform hypotheses has already been highlighted in the domain of information retrieval by Fuhr and Rolleke who propose to modify the program in order to simulate the combination of the open world and closed world assumptions \[9\].
has to decide whether to charge a person named Ted accused of murder. To do so, the judge first collects facts from two different sources: the public prosecutor and the person’s lawyer. The judge then combines the collected facts using a set of rules in order to reach a decision. For the sake of our example, let us suppose that the judge has collected a set of facts $F$ that he combines using a set of rules $R$ as follows:

$$F = \{\text{witness(John), friends(John,Ted)}\}$$

$$R = \begin{cases} 
\text{suspect(X) \leftarrow motive(X)} \\
\text{suspect(X) \leftarrow witness(X)} \\
\text{innocent(X) \leftarrow alibi(X,Y) \land \neg friends(X,Y)} \\
\text{innocent(X) \leftarrow presumption\_of\_innocence(X) \land \neg suspect(X)} \\
\text{friends(X,Y) \leftarrow friends(Y,X)} \\
\text{friends(X,Y) \leftarrow friends(X,Z) \land friends(Z,Y)} \\
\text{charge(X) \leftarrow suspect(X)} \\
\text{charge(X) \leftarrow \neg innocent(X)} \\
\end{cases}$$

Some comments on the rules. The two first rules of $R$ describe how the prosecutor works: in order to support the claim that a person $X$ is a suspect, the prosecutor tries to provide a motive (first rule) or a witness against $X$ (second rule). The third and fourth rules of $R$ describe how the lawyer works: in order to support the claim that $X$ is innocent, the lawyer tries to provide an alibi for $X$ by a person who is not a friend of $X$ (third rule), or to use the presumption of innocence if the person is not suspect (fourth rule), i.e. the defendant is assumed innocent until proved guilty. Finally, the last two rules of $R$ are the judge’s “decision making rules”.

What should the value of $\text{charge(Ted)}$ be? $\text{Ted}$ should be charged if all the attempts in deriving the innocence of $\text{Ted}$ have failed or if the prosecutor has succeeded in proving that $\text{Ted}$ is effectively suspect.

We can easily remark that uniform hypotheses, as the closed word assumption and the open world assumption, do not fit with our expectation. If we follow the Closed World Assumption (CWA) and assign to all atoms the default value false, then the judge will infer that $\text{Ted}$ is not innocent and must be charged. If we follow the OWA and assign to all atoms the default value unknown, then the atoms $\text{suspect(Ted), innocent(Ted)}$ and $\text{charged(Ted)}$ are unknown.

---

3 For ease of presentation we leave uncertainties out. For instance, there might well be some uncertainty in the facts and rules, as we will see later on.
and the judge cannot take a decision (that semantics is often considered as too weak). We could consider another uniform hypothesis, i.e. the assumption that assigns to all atoms the default value true, but under such an assumption, the judge will infer that Ted is suspect and must be charged.

An intuitively appealing non-uniform hypothesis in this situation is to assume by default that the atoms motive(Ted), witness(Ted) and suspect(Ted) are false, that the atom presumption_of_innocence(Ted) is true and that the others are unknown. With such a hypothesis, the judge could infer that Ted is innocent, not suspect and should not be charged.

We believe that we should not be limited to consider fixed assumptions only, but be able to associate to a logic program a semantics based on any given hypothesis, which represents our default or assumed knowledge. To this end, in this paper we will extend the parametric IB framework [16], an unifying umbrella for IB frameworks, along three directions: (i) we will introduce negation into the programs, i.e. we will extend the IB frameworks with the non-monotonic mode of negation; (ii) the default truth values of atoms have not necessarily to be all equal each other; and (iii) a hypothesis can be a partial interpretation, i.e. we do not require that every atom has a default truth value (an atom’s truth value may be still unknown).

We will show that our approach extends the usual ones: if we restrict our attention to logic programs and consider total uniform hypotheses, then our semantics reduces to the usual semantics of logic programs. In particular, we will show that under the everywhere false assumption (i) for programs without negation we obtain the semantics presented in [16]; and (ii) for Datalog programs with negation we obtain the Well Founded Semantics (WFS) [23]. On the other hand, under the empty hypothesis (the atom’s default truth is unknown) our semantics includes the Kripke-Kleene semantics of Fitting [5].

In the following we proceed as follows. In the next section we introduce the syntax of our logical language, we define the notion of satisfiability and present fixpoint operators, through which in Section 3 the intended semantics of our logic programs is specified. Section 4 compares our semantics with others, while Section 5 concludes.

2 Syntax and semantics: preliminaries

We recall the syntactical aspects of the parametric IB framework presented in [16] and extend it with negation.

Let \( \mathcal{L} \) be an arbitrary first order language that contains infinitely many vari-
able symbols, finitely many constants, and predicate symbols, but no function symbols. While $L$ does not contain function symbols, it contains symbols for families of propagation ($F_p$), conjunction ($F_c$) and disjunction functions ($F_d$), called combination functions.

Let $\langle T, \preceq, \otimes, \oplus \rangle$ be a certainty lattice (a complete lattice) and $B(T)$ the set of finite multisets over $T$. With $\bot$ and $\top$ we denote the least and greatest element in $T$, respectively. A propagation function is a mapping from $T \times T$ to $T$ and a conjunction or disjunction function is a mapping from $B(T)$ to $T$. Each kind of function must verify some of the following properties:

1. monotonicity w.r.t. (with respect to) each one of its arguments;
2. continuity w.r.t. each one of its arguments;
3. bounded-above: $f(\alpha_1, \alpha_2) \preceq \alpha_i$, for $i = 1, 2, \forall \alpha_1, \alpha_2 \in T$;
4. bounded-below: $f(\alpha_1, \alpha_2) \succeq \alpha_i$, for $i = 1, 2, \forall \alpha_1, \alpha_2 \in T$;
5. commutativity: $f(\alpha_1, \alpha_2) = f(\alpha_2, \alpha_1)$, $\forall \alpha_1, \alpha_2 \in T$;
6. associativity: $f(\alpha_1, f(\alpha_2, \alpha_3)) = f(f(\alpha_1, \alpha_2), \alpha_3)$, $\forall \alpha_1, \alpha_2, \alpha_3 \in T$;
7. $f(\{\alpha\}) = \alpha$, $\forall \alpha \in T$;
8. $f(\emptyset) = \bot$;
9. $f(\emptyset) = \top$;
10. $f(\alpha, \top) = \alpha$, $\forall \alpha \in T$.

As postulated in [16], we require that:

1. any conjunction function in $F_c$ satisfies properties 1, 2, 3, 5, 6, 7, 9 and 10;
2. any propagation function in $F_p$ satisfies properties 1, 2 and 10;
3. any disjunction function in $F_d$ satisfies properties 1, 2, 4, 5, 6, 7 and 8.

We also assume that there is a function from $T$ to $T$, called negation function and denoted $\neg$, that is anti-monotone w.r.t. $\preceq$ and satisfies $\neg(\neg \alpha) = \alpha$, $\forall \alpha \in T$ and $\neg(\bot) = \top$.

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

1. $T$ is a set of truth values partially ordered by $\preceq$. We assume that $\langle T, \preceq, \otimes, \oplus \rangle$ is a complete lattice, where $\otimes$ is the meet operator and $\oplus$ the join operator;
2. $R$ is a finite set of normal parametric rules (np-rules), each of which is

---

4 For simplicity, we formulate the properties treating any function as a binary function on $T$. 

5
a statement of the form:

\[ r : A \overset{\alpha_r}{\leftarrow} B_1, ..., B_n, \neg C_1, ..., \neg C_m \]

where \( A \) is an atomic formula and \( B_1, ..., B_n, C_1, ..., C_m \) are atomic formulas or values in \( T \) and \( \alpha_r \in T \setminus \{ \bot \} \) is the certainty of the rule;

(3) \( C \) is a mapping, which associates with each np-rule a conjunction function in \( \mathcal{F}_c \);

(4) \( P \) is a mapping, which associates with each np-rule a propagation function in \( \mathcal{F}_p \);

(5) \( D \) is a mapping, which associates with each predicate symbol in \( P \) a disjunction function in \( \mathcal{F}_d \).

For ease of presentation, we write

\[ r : A \overset{f_d \in \mathcal{F}_d}{\leftarrow} B_1, ..., B_n, \neg C_1, ..., \neg C_m; (f_d, f_p, f_c) \]

to represent a np-rule in which \( f_d \in \mathcal{F}_d \) is the disjunction function associated with the predicate symbol \( A \) and, \( f_c \in \mathcal{F}_c \) and \( f_p \in \mathcal{F}_p \) are respectively the conjunction and propagation functions associated with \( r \). The intention is that the conjunction function (e.g. \( \otimes \)) determines the truth value of the conjunction of \( B_1, ..., B_n, \neg C_1, ..., \neg C_m \), the propagation function (e.g. \( \otimes \)) determines how to “propagate” the truth value resulting from the evaluation of the body to the head, by taking into account the certainty \( \alpha_r \) associated to the rule \( r \), while the disjunction function (e.g. \( \oplus \)) dictates how to combine the certainties in case an atom is head of several rules. Note that in case negation is missing, np-programs are parametric programs (p-programs) as defined in [16].

We further define the Herbrand base \( \mathcal{H}_P \) of a np-program \( P \) as the set of all instantiated atoms corresponding to atoms appearing in \( P \) and define \( P^\ast \) to be the Herbrand instantiation of \( P \), i.e. the set of all ground instantiations of the rules in \( P \).

A classical logic program is a np-program such that \( \otimes \) is the unique conjunction and propagation function, \( \oplus \) is the unique disjunction function and \( \alpha_r = \top \), for all rule \( r \in P \). Such a program will be denoted in the classical way.

Example 2 Consider the complete lattice \( \langle T, \preceq, \otimes, \oplus \rangle \), where \( T = [0, 1] \), \( \forall a, b \in [0, 1], a \preceq b \) iff \( a \leq b \), \( a \otimes b = \min (a, b) \), and \( a \oplus b = \max (a, b) \). Consider the disjunction function \( f_d (\alpha, \beta) = \alpha + \beta - \alpha \cdot \beta \), the conjunction function \( f_c (\alpha, \beta) = \alpha \cdot \beta \) and the propagation function \( f_p = f_c \). The negation function is the usual function \( \neg (\cdot) = 1 - (\cdot) \). Then the following is a
np-program $P$:\footnote{The symbol $\neg$ instead of a function denotes the facts that this function is not relevant. Note that any conjunction function is also a propagation function.}

$$P = \begin{cases} \text{suspect}(X) & \overset{0.6}{\leftarrow} \text{motive}(X) \langle f_d, \otimes, - \rangle \\ \text{suspect}(X) & \overset{0.8}{\leftarrow} \text{witness}(X) \langle f_d, \otimes, - \rangle \\ \text{innocent}(X) & \overset{1}{\leftarrow} \text{alibi}(X,Y) \land -\text{friends}(X,Y) \langle f_d, f_p, \otimes \rangle \\ \text{innocent}(X) & \overset{1}{\leftarrow} \text{presumption of innocence}(X) \land -\text{suspect}(X) \langle f_d, f_p, \otimes \rangle \\ \text{friends}(X,Y) & \overset{1}{\leftarrow} \text{friends}(Y,X) \langle \oplus, f_p, - \rangle \\ \text{friends}(X,Y) & \overset{0.7}{\leftarrow} \text{friends}(X,Z) \land \text{friends}(Z,Y) \langle \oplus, f_p, f_c \rangle \\ \text{charge}(X) & \overset{1}{\leftarrow} \text{suspect}(X) \langle \oplus, f_p, - \rangle \\ \text{charge}(X) & \overset{1}{\leftarrow} -\text{innocent}(X) \langle \oplus, f_p, - \rangle \\ \text{witness}(\text{John}) & \overset{1}{\leftarrow} 1 \langle \oplus, f_p, - \rangle \\ \text{motive}(\text{Jim}) & \overset{1}{\leftarrow} 0.8 \langle \oplus, f_p, - \rangle \\ \text{alibi}(\text{Jim}, \text{John}) & \overset{1}{\leftarrow} 1 \langle \oplus, f_p, - \rangle \\ \text{friends}(\text{John}, \text{Ted}) & \overset{1}{\leftarrow} 0.8 \langle \oplus, f_p, - \rangle \\ \text{friends}(\text{Jim}, \text{Ted}) & \overset{1}{\leftarrow} 0.6 \langle \oplus, f_p, - \rangle \end{cases}$$

Note that e.g. for predicate \text{suspect}, the disjunction function $f_d$ is associated, as if there are different ways to infer that someone is suspect, then we would like to increase our suspicion and not just to choose the maximal value. \hfill $\Box$

### 2.1 Interpretations of programs

An interpretation of a np-program $P$ is a function that assigns to all atoms of the Herbrand base of $P$ a value in $T$. We denote $\mathcal{V}_P(T)$ the set of all interpretations of $P$. Of course, an important issue is to determine which is the intended meaning or semantics of a np-program. Following the usual approach, the semantics of a program $P$ is determined by selecting a particular interpretation of $P$ in the set of models of $P$. In logic programs without negation, as well as in the parametric IB framework, that chosen model is usually the least model of $P$ w.r.t. $\preceq$.

Introducing negation in classical logic programs, and in particular in our parametric IB framework, has as consequence that some np-programs do not have an unique minimal model, as shown in the following example.
Example 3 Let $T$ be $[0, 1]$. Consider, as usual, $f_c(\alpha, \beta) = \min(\alpha, \beta)$, $f_d(\alpha, \beta) = \max(\alpha, \beta)$, $f_p(\alpha, \beta) = \alpha \cdot \beta$ and the usual negation function. Consider the program $P = \{ A \leftarrow \neg B; \langle f_d, f_p, - \rangle, B \leftarrow \neg A; \langle f_d, f_p, \rangle, A \leftarrow 0.2; \langle f_d, f_p, - \rangle, B \leftarrow 0.3; \langle f_d, f_p, \rangle \}$. This program will have an infinite number of models $I_x^y$, where $0.2 \leq x \leq 1$, $0.3 \leq y \leq 1$, $y \geq 1 - x$, $I_x^y(A) = x$ and $I_x^y(B) = y$ (those in the grey zone in Figure 1). There are also an infinite number of minimal models (those on the bold line) according to the order $(a, b) \preceq (c, d)$ iff $a \preceq c$ and $b \preceq d$. These minimal models $I_x^y$ are such that $y = 1 - x$. □

An usual way to deal with such situations in classical logic consists in considering partial interpretations. Indeed, we will consider multi-valued partial interpretations, i.e. interpretations that assign values only to some atoms of $\mathcal{HBP}$ and are not defined for the other atoms. An atom that is not defined in a partial interpretation can be seen as an underdefined atom: that atom may have a value in $T$ but that value is currently unknown.

Definition 2 (Partial interpretation) Let $P$ be a np-program. A partial interpretation $I$ of $P$ is a set $\{ A : \mu \mid A \in \mathcal{HBP} \text{ and } \mu \in T \}$.

A partial interpretation $I$ can be seen as a function defined by: for all ground atoms $A$, if $A : \mu \in I$ then $I(A) = \mu$ else $I(A)$ is not defined. Of course, an interpretation is a partial interpretation. Interpretations and partial interpretations will be used as functions or as sets following the context.

In the following, given a np-program $P$, (i) we denote with $r_A$ a rule $(r : A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m; \langle f_d, f_p, f_c \rangle) \in P^*$, whose head is $A$; (ii) given an interpretation $I$ such that each premise in the body of $r_A$ is defined under $I$, with $I(r_A)$ we denote the evaluation of the body of $r_A$ w.r.t. $I$, i.e.

$$I(r_A) = f_p(\alpha_r, f_c(I(B_1), \ldots, I(B_n), \neg I(C_1), \ldots, \neg I(C_m)))$$

and (iii) $I(r_A)$ is undefined in case some premise in the body is undefined in $I$, except for the case where there is an $i$ such that $I(B_i) = \perp$ or $I(C_i) = \top$. In that case, we define $I(r_A) = \perp$.

Definition 3 (Satisfaction of a np-program) Let $P$ be a np-program and let $I$ be a partial interpretation of $P$. Then we say that $I$ satisfies (is a model
of) \( P \), denoted \( \models_I P \), iff \( \forall A \in \mathcal{H}B_P: \)

1. if there is a rule \( r_A \in P^* \) such that \( I(r_A) = \top \), then \( I(A) = \top \);
2. for some rule \( r_A \in P^* \), \( I(r_A) \) is undefined and case 1. does not hold;
3. if for all rules \( r_A \in P^* \), \( I(r_A) \) is defined, then \( I(A) \geq f_d(X) \), where

\[
X = \{ I(r_A) : r_A \in P^* \}.
\]

\( f_d \) is the disjunction function associated with \( \pi(A) \), the predicate symbol of \( A \).

**Example 4** For the program \( P \) in Example 3, the interpretations \( I_A \) are all models of \( P \). Note that the interpretation \( I \) undefined on \( A \) and \( B \) is a model of \( P \) as well.

It is worth noting that if we restrict our attention to positive programs only, then the definition of satisfiability of np-programs reduces to that of satisfiability of p-programs defined in [16], where the interpretation \( I \) is not partial but defined for all atoms in \( \mathcal{H}B(P) \).

2.2 Parameterized Operators

First, we extend the ordering on \( \mathcal{T} \) to the space of interpretations \( \mathcal{V}_P(\mathcal{T}) \). Let \( I_1 \) and \( I_2 \) be in \( \mathcal{V}_P(\mathcal{T}) \), then \( I_1 \preceq I_2 \) if and only if \( I_1(A) \preceq I_2(A) \) for all ground atoms \( A \). Under this ordering \( \mathcal{V}_P(\mathcal{T}) \) becomes a complete lattice, and we have \( (I_1 \otimes I_2)(A) = I_1(A) \otimes I_2(A) \), and similarly for the other operators. The actions of functions can be extended from atoms to formulae as follows: \( I(f_c(X,Y)) = f_c(I(X),I(Y)) \), and similarly for the other functions. Finally, for all \( \alpha \) in \( \mathcal{T} \) and for all \( I \) in \( \mathcal{V}_P(\mathcal{T}) \), \( I(\alpha) = \alpha \).

We now define a new operator \( T^H_P \) inspired by [7,17,24–26]. That operator is parameterized by an interpretation on \( \{\bot, \top\} \). That interpretation represents our default knowledge and we will call it a hypothesis to stretch the fact that it represents default knowledge and not “sure knowledge”. Such a hypothesis asserts that some atoms are assumed \( \bot \) (false) and some others are assumed to be \( \top \) (true). The operator \( T^H_P \) infers new information from two interpretations: the first one is used to evaluate the positive literals, while the second one is used to evaluate the negative literals of the bodies of rules in \( P \).

**Definition 4 (Parameterized immediate consequence operator)** Let \( P \) and \( H \) be any np-program and a hypothesis, respectively. The immediate consequence operator \( T^H_P \) is a mapping from \( \mathcal{V}_P(\mathcal{T}) \times \mathcal{V}_P(\mathcal{T}) \) to \( \mathcal{V}_P(\mathcal{T}) \), defined as follows: for every pair \( (I, J) \) of interpretations in \( \mathcal{V}_P(\mathcal{T}) \), for every atom \( A \), if there is no rule in \( P \) with \( A \) as its head, then \( T^H_P(I, J)(A) = H(A) \), else \( T^H_P(I, J)(A) = f_d(X) \), where \( f_d \) is the disjunction function associated with
\( \pi(A) \), the predicate symbol of \( A \), and

\[
X = \{ f_p(\alpha, f_c(\{I(B_1), \ldots, I(B_n), \neg J(C_1), \ldots, \neg J(C_m)\})) : \\
\{ \langle f_d, f_{pr}, f_c \rangle \in P^* \} \} .
\]

Note that in case negation is absent and \( H \) assigns the value \( \perp \) to all the atoms, then \( T_H^P \) reduces to the immediate consequence operator defined in [16].

**Proposition 1** Let \( P \) and \( H \) be any np-program and a hypothesis, respectively. \( T_H^P \) is monotonic in its first argument, and anti-monotonic in its second argument w.r.t. \( \preceq \).

Using Proposition 1 and the Knaster-Tarsky theorem, we can define an operator \( S_H^P \), likewise [10] and derived from \( T_H^P \), that takes an interpretation \( J \) as input, first evaluates the negative literals of the program w.r.t. \( J \), and then returns the model of the resulting “positive” np-program obtained by iterations of \( T_H^P \) beginning with hypothesis \( H \).

But, in order to deal with non-uniform hypotheses, at first we need to define the notion of stratification w.r.t. positive cycles and then activate the rules w.r.t. that stratification.

**Definition 5 (Extended positive cycle)** A positive cycle of a np-program \( P \) is a set \( \{r_{A_1}, \ldots, r_{A_n}\} \) of rules of \( P \) such that for all \( i \) in \([1\ldots n]\), \( A_i \) appears positively in the body of \( r_{A_{i+1}} \) and \( A_n \) appears positively in the body of \( r_{A_1} \). A positive cycle \( C \) extended with all the rules in \( P \) whose head is the head of one of the rules of \( C \) is called extended positive cycle.

**Definition 6 (Stratification w.r.t. extended positive cycles)** A stratification w.r.t. extended positive cycles of a np-program \( P \) is a sequence of np-programs \( P_1, \ldots, P_n \) such that for the mapping \( \sigma \) from rules of \( P \) to \([1\ldots n]\),

(1) \( P_1, \ldots, P_n \) is a partition of \( P \);
(2) every rule \( r \) is in \( P_{\sigma(r)} \);
(3) if \( r_1 \) and \( r_2 \) are two rules of \( P \) such that the head of \( r_2 \) appears in the body of \( r_1 \) and there is no extended positive cycle of rules of \( P \) containing \( r_1 \), then \( \sigma(r_1) = \sigma(r_2) \);
(4) if \( r_1 \) and \( r_2 \) are two rules of \( P \) appearing in the same extended positive cycle of rules of \( P \), then \( \sigma(r_1) = \sigma(r_2) \);
(5) if \( r_1 \) and \( r_2 \) are two rules of \( P \) such that the head of \( r_2 \) appears in the body of \( r_1 \) and there is an extended positive cycle of rules of \( P \) containing \( r_1 \) but no extended positive cycle of rules of \( P \) containing both \( r_1 \) and \( r_2 \), then \( \sigma(r_2) < \sigma(r_1) \).
We can remark that every np-program has a stratification w.r.t. extended positive cycles. It can be shown that

**Proposition 2** Let $P$ be a np-program that has a stratification w.r.t. extended positive cycles containing only one stratum, and let $J$ be an interpretation over $T$. Let $H$ be a partial hypothesis over $\{\bot, \top\}$ such that for all extended positive cycles $\{r_{A_1}, ..., r_{A_k}\}$ of $P$, $H(A_1) = ... = H(A_k)$. Then the sequence defined by $a_0 = H$ and $a_{n+1} = T_H^P(a_n, J)$ converges. 

We now define the operator $S_H^P$ derived from $T_H^P$.

**Definition 7 (The parameterized alternating operator $S_H^P$)** Let $P$ be a np-program that has a stratification w.r.t. the extended positive cycles $P_1, ..., P_n$ and let $J$ be an interpretation over $T$. Let $H$ be an interpretation over $\{\bot, \top\}$ such that for all extended positive cycles $\{r_{A_1}, ..., r_{A_k}\}$ of $P$, $H(A_1) = ... = H(A_k)$. Then $S_H^P(J)$ is the limit of the following sequence:

- $a_1$ is the iterated fixpoint of the function $\lambda x. T_{P_1}^H(x, J)$ obtained when beginning the computation with $H$;
- $a_i$ is the iterated fixpoint of the function $\lambda x. T_{P_1 \cup ... \cup P_i}^H(x, J)$ obtained when beginning the computation with $a_{i-1}$.

Intuitively, during that computation of $S_H^P(J)$, we fix the value of the negative premises in $P$ with their values in $J$. Then we consider the “positive program” and evaluate that program stratum by stratum. After the evaluation of a stratum, we know that the knowledge obtained cannot be modified by what we will infer by activating the rules of the next strata. Then we use that knowledge to evaluate the next stratum. Generally, a program may have more than one stratification w.r.t. extended positive cycles. However, the result of the computation does not depend on the stratification used for the computation. Note that the notion of stratification and the condition on $H$ that we have introduced are indispensable for the convergence of the computation. To illustrate this point, consider the following examples.

**Example 5** Let $H = \{A : \top, B : \bot, C : \bot, D : \bot\}$ be a hypothesis and let $P$ be a np-program defined by $P = \{A \leftarrow \bot, B \leftarrow \bot, C \leftarrow A, D \leftarrow B, C \leftarrow D, D \leftarrow C\}$. If we compute the sequence $I_0 = H, I_i = T_H^P(I_{i-1}, J)$ then we have

- $I_1 = \{A : \bot, B : \bot, C : \top, D : \bot\}$,
- $I_2 = \{A : \bot, B : \bot, C : \bot, D : \top\}$,
- $I_3 = \{A : \bot, B : \bot, C : \top, D : \bot\} \ldots$

This computation does not terminate. Now, if we consider our definition of
then we have a stratification of \( P \) with two strata: the first one contains the two first rules of \( P \) and the second one the four last rules of \( P \). The computation terminates, and we have \( I_1 = \{ A : \perp, B : \perp, C : \perp, D : \perp \} = I_2 \).
\]

\[ \]

\noindent \textbf{Example 6} Let \( P \) be a np-program \( P = \{ A \leftarrow B, B \leftarrow A \} \) and \( H \) the assumption \( H = \{ A : \top, B : \perp \} \). The condition on \( H \) is not satisfied, i.e. \( H(A) \neq H(B) \), and the computation does not terminate.
\]

\[ \]

\section{Semantics under non-uniform assumptions}

In this section we will determining which, among all the models, is the intended model of a np-program w.r.t. a given hypothesis. For the rest of the paper, if not stated otherwise, any hypothesis is supposed to assign the same default value to the atoms that are heads of rules of a same extended positive cycle.

\[ \]

\subsection{Semantics under total non-uniform assumptions}

From Proposition 1, we derive the following property of \( S_P^H \).

\[ \]

\textbf{Proposition 3} Let \( P \) and \( H \) be any np-program and a total non-uniform hypothesis, respectively. Then \( S_P^H \) is anti-monotone w.r.t. \( \leq \) and, thus \( S_P^H \circ S_P^H \) is monotone.

\[ \]

There is a well-know property, which derives from the Knaster-Tarski theorem and deals with anti-monotone functions on complete lattices:

\[ \]

\textbf{Proposition 4 ([32])} Suppose that a function \( f \) is anti-monotone on a complete lattice \( T \). Then there are unique two elements \( \mu \) and \( \nu \) of \( T \), called extreme oscillation points of \( f \), such that the following hold:

\[ \]

- \( \mu \) and \( \nu \) are the least and greatest fixpoint of the composition \( f \circ f \);
- \( f \) oscillates between \( \mu \) and \( \nu \) in the sense that \( f(\mu) = \nu \) and \( f(\nu) = \mu \);
- if \( x \) and \( y \) are also elements of \( T \) between which \( f \) oscillates then \( x \) and \( y \) lie between \( \mu \) and \( \nu \).

\[ \]

Under the ordering \( \preceq \), \( S_P^H \) is anti-monotone and \( \mathcal{V}_P(T) \) is a complete lattice, so \( S_P^H \) has two extreme oscillation points under this ordering. Let \( I_\perp \) be the interpretation that assigns the value \( \perp \) to all atoms of \( \mathcal{H}B(P) \), i.e. the minimal element of \( \mathcal{V}_P(T) \) w.r.t. \( \preceq \), while let \( I_\top \) be the interpretation that assigns the value \( \top \) to all atoms of \( \mathcal{H}B(P) \), i.e. the maximal element of \( \mathcal{V}_P(T) \) w.r.t. \( \preceq \).
Proposition 5 Let \( P \) and \( H \) be any np-program and a total non-uniform hypothesis, respectively. \( S_H^H \) has two extreme oscillation points, \( S_H^H = (S_P^H \circ S_P^H)\infty(I_\perp) \) and \( S_H^H = (S_P^H \circ S_P^H)\infty(I_\top) \), with \( S_H^H \preceq S_H^H \).

Similarly to van Gelder’s alternated fixpoint approach \[10\], \( S_H^\perp \) and \( S_H^\top \) are respectively an under-estimation and an over-estimation of \( P \), but w.r.t. any hypothesis \( H \). As the meaning of \( P \) we propose to consider as defined only the atoms whose values coincide in both limit interpretations. It can be shown that

Proposition 6 Let \( P \) and \( H \) be any np-program and a total non-uniform hypothesis, respectively. Then \( \models S_H^\perp \cap S_H^\top \).

The interpretation \( S_H^\perp \cap S_H^\top \) is a model of \( P \), and will be the intending meaning or semantics of \( P \) w.r.t. the assumption \( H \).

Definition 8 (Compromise semantics) Let \( P \) be any np-program, the compromise semantics of \( P \) w.r.t. a total non-uniform hypothesis \( H \), \( CS_H^H(P) \), is defined by \( CS_H^H(P) = S_H^\perp \cap S_H^\top \).

Example 7 Let \( P \) be the np-program of Example 2 and \( I_\perp \), then we have\[7\] \( CS^{I_\perp}(P) \supset \{s(John):0.8, s(Jim):0.6, s(Ted):0, i(John):0, i(Jim):0.664, i(Ted):0, c(John):1, c(Jim):0.6, c(Ted):1\} \). Now let \( H \) be the following hypothesis \( H = \{m(X):0, w(x):0, s(X):0, p(X):1, a(X,Y):0, f(X,Y):0, i(X):1, c(X):0\} \). Then we have \( CS^H(P) \supset \{s(John):0.8, s(Jim):0.6, s(Ted):0, i(John):0.2, i(Jim):0.7984, i(Ted):1, c(John):0.8, c(Jim):0.6, c(Ted):0\} \).

3.2 Semantics under partial non-uniform assumptions

In the previous section we dealt with total non-uniform assumptions. In this section we address the case where assumptions may be partial.

A possible idea to deal with such a situation is to introduce a new logical value \( u \) and define a new lattice \( T' = T \cup \{u\} \). That value represents an unknown or undefined value that means that \( u \) is used to replace a value in \( T \) that is currently not known. We would like to extend conjunction and disjunction functions in the following way:

- \( f_c(u,x) = \text{if } x \neq \perp \text{ then } u \text{ else } \perp \);
• \( f_d(u, x) = \) if \( x \neq \top \) then \( u \) else \( \top \).

We need also to extend the order \( \preceq \) in \( T \) to \( T' \). We know that for all \( x \in T' \), \( \bot \preceq x \preceq \top \). But, from the first constraint it follows that \( u \preceq x \) for all \( x \neq \bot \) and, similarly, from the second one it follows that \( x \preceq u \) for all \( x \neq \top \).

But then, there is a solution only if \( T = \{ \bot, \top \} \). We follow another way: a partial hypothesis \( H \) on \( \{ \bot, \top \} \) can be seen as the intersection between two total interpretations \( H_\bot \) and \( H_\top \), where \( H_\bot \) is as \( H \) except that \( \bot \) is assumed for the unknown atoms, while \( H_\top \) is as \( H \) except that \( \top \) is assumed for the unknown atoms. Note that \( H_\bot \cap H_\top = H \). In order to assign a semantics to a np-program w.r.t. such a partial interpretation, we propose to consider the intersection or consensus between the two semantics. It can be shown that

**Proposition 7** Let \( P \) be any np-program and let \( H \) be a partial non-uniform hypothesis. It follows that \( \models_{CS^H(P) \cap CS^{I_\bot}(P)} P \).

That intersection is effectively a model of \( P \) and will be the meaning of \( P \).

**Definition 9 (Consensus semantics w.r.t. \( H \))** Let \( P \) be any np-program and let \( H \) be a partial non-uniform hypothesis. The consensus semantics of \( P \) w.r.t. \( H \), \( C^H(P) \), is defined by

\[
C^H(P) = CS^{H_\bot}(P) \cap CS^{I_\bot}(P).
\]

**Example 8** Let \( P \) be the np-program of Example 2 and \( \bot \) the hypothesis suggested in the introduction, i.e. \( \bot \). Then we have \( C^H(P) \supset \{ s(\text{John}) : 0.8, s(\text{Jim}) : 0.6, s(\text{Ted}) : 0, i(\text{Ted}) : 1, c(\text{John}) : 0.8, c(\text{Jim}) : 0.6, c(\text{Ted}) : 0 \} \).

4 Comparisons with usual semantics

Our semantics extends the Lakshmanan and Shiri’s semantics [16] of parametric programs to normal parametric programs. This is due to the fact that the machinery developed in order to deal with negation has no effect on positive programs, thus for the everywhere false hypothesis \( I_\bot \), we have

**Proposition 8** If \( P \) is a np-program without negation then the compromise semantics \( CS^{I_\bot}(P) \) of \( P \) (or equivalently the consensus semantics \( C^{I_\bot}(P) \)) w.r.t. the hypothesis \( I_\bot \) coincides with the Lakshmanan and Shiri’s semantics of \( P \).

Now, we compare our semantics with the well-founded semantics of classical logic programs defined in [23].

\[\text{Note that any compromise semantics is also a consensus semantics.}\]
Proposition 9 Let \( P \) be a Datalog program with negation. The compromise semantics \( CS^{I_{\perp}}(P) \) of \( P \) (or equivalently the consensus semantics \( C^{I_{\perp}}(P) \)) w.r.t. the hypothesis \( I_{\perp} \) coincides with the well-founded semantics of \( P \). \hfill \Box

It is worth noting that our approach extends the well-founded semantics to the IB framework as well.

Example 9 Consider Example 2 and hypothesis \( I_{\perp} \). We define the Datalog program with negation \( P' \) by replacing in \( P \) all the truth values by 1, the disjunction function \( f_d \) by \( \oplus \) and the conjunction function \( f_c \) by \( \otimes \). Then we have \( C^{I_{\perp}}(P') = \{ s(\text{John}):1, s(\text{Jim}):1, s(\text{Ted}):0, i(\text{John}):0, i(\text{Jim}):0, i(\text{Ted}):0, c(\text{John}):1, c(\text{Jim}):1, c(\text{Ted}):1 \} \).

Finally, we can compare the semantics corresponding to the empty set hypothesis with the usual semantics and in particular with the Kripke-Kleene semantics [5].

Proposition 10 Let \( P \) be a Datalog program with negation and \( H_{\emptyset} \) be the empty hypothesis. Let \( WFS(P) \) be the well-founded semantics of \( P \) and \( KK(P) \) be the Kripke-Kleene semantics of \( P \). Then \( KK(P) \subset C^{H_{\emptyset}}(P) \subset WFS(P) \) holds. \hfill \Box

The consensus semantics \( C^{H_{\emptyset}}(P) \) of \( P \) w.r.t. the empty hypothesis \( H_{\emptyset} \) represents more knowledge than the Kripke-Kleene semantics of \( P \), but less than the well-founded semantics of \( P \). This results show the well-known fact that the Kripke-Kleene semantics of \( P \) is weaker than the well-founded semantics of \( P \). Moreover, our semantics seems to be a good compromise between both those semantics: the Kripke-Kleene semantics of \( P \) may be considered as too weak, while the well-founded semantics of \( P \), may be considered as too strong, as shown in the following examples.

Example 10 For \( P = \{ A \leftarrow B, A \leftarrow \neg B \} \), \( KK(P) = \emptyset \subset C^{H_{\emptyset}}(P) = \{ A : \top \} \subset WFS(P) = \{ A : \top, B : \bot \} \) holds, while for \( P = \{ A \leftarrow B \land \neg B \} \), \( KK(P) = \emptyset \subset C^{H_{\emptyset}}(P) = \{ A : \bot \} \subset WFS(P) = \{ A : \bot, B : \bot \} \) holds. \hfill \Box

5 Conclusion

We have seen a general framework for reasoning about uncertainty and negation in deductive databases and logic programming. Our framework uses parameterized semantics of implication-based logic programming with negation under non-uniform assumptions for the missing information. We have also seen that when we restrict our framework to uniform assumptions only, our approach captures the semantics of conventional logic programs.
References


