Paraconsistent Logic Programs

João Alcântara, Carlos Viegas Damásio and Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Universidade Nova de Lisboa
2829-516 Caparica, Portugal. E-mail: `jfla|cd|lmp@di.fct.unl.pt`

Abstract. We propose a framework which extends Antitonic Logic Programs [3] to an arbitrary complete bilattice of truth-values, where belief and doubt are explicitly represented. Based on Fitting's ideas, this framework allows a precise definition of important operators found in logic programming such as explicit negation and the default negation. In particular, it leads to a natural integration of explicit negation with the default negation through the coherence principle [20]. According to this principle, the explicit negation entails the default negation. We then define Coherent Answer Sets, and the Paraconsistent Well-founded Model semantics, generalizing paraconsistent semantics for logic programs (for instance, $WFSX_p$ [2]). Our framework is an extension of Antitonic Logic Programs in the most cases, and is general enough to capture Probabilistic Deductive Databases, Possibilistic Logic Programming, Hybrid Probabilistic Logic Programs, and Fuzzy Logic Programming. Thus, we have a powerful mathematical formalism for dealing with default reasoning, paraconsistency, and uncertainty. We illustrate its adumbration of paraconsistency with an embedding of $WFSX_p$ into our framework.

1 Introduction

In complex applications of knowledge representation we may be faced with uncertainty, incompleteness, and inconsistency. As result, several frameworks for manipulating data and knowledge have been proposed in the form of extensions to classical logic programming and deductive databases. Usually, the authors characterize their programs with a model theoretical semantics, where a minimum model is guaranteed to exist, and a corresponding monotonic fixpoint operator too (whether continuous or not).

The underlying uncertainty formalism in the proposed logic programming frameworks includes probability theory [15, 18], fuzzy set theory [21, 22], many-valued logic [7, 8, 13], possibilistic logic [6]. Furthermore, different ways of dealing with uncertainty may be required for one given application [16]. All these logic programming based frameworks are monotonic, and none allows default negation.

Following an algebraic approach to both the language and the semantics of logic programs, Damásio and Pereira define in [4] a rather general framework of Monotonic Logic Programs, where the rules are constituted by arbitrary isotonic body functions and by propositional symbols in the head. These programs extend definite logic programming, i.e. those without non-monotonic or default negation, to arbitrary complete lattices of truth-values with an appropriate notion of implication. It is also shown that they are general enough to capture several distinct logic programming semantics such as the uncertainty formalisms above. In [3], the same authors generalize the framework to allow for rules with arbitrary antitonic bodies over general complete lattices, and show that all the standard logic programming theoretical results carry over to such Antitonic Logic Programs, and define for them Stable and Well-founded Model semantics alike.

Notwithstanding, a specific treatment for the explicit negation in Antitonic Logic Programs is not provided. In this work we extend the framework for these programs to an arbitrary complete bilattice of truth-values, where both belief and doubt are explicitly represented. We dub the resultant framework Paraconsistent Logic Programs. In its presentation, we are motivated by Fitting's bilattices [8], and by Lakshmanan and Sadri's work on probabilistic deductive databases [15]. Fitting's bilattices support an elegant framework for logic programming involving belief and doubt. In particular, they lead to a precise definition of explicit negation operators. We further use these results to properly characterize default negation and to ensure obedience to the coherence principle, which states negation entails default negation, i.e., anything that is explicitly false must be default false as well. On the other hand, in [15], the authors argue about the convenience of explicitly representing both belief and doubt when dealing with incomplete knowledge, where different evidence may contradict one another.

Besides satisfying the coherence principle, a semantics for Paraconsistent Logic Programs must deal with both contradiction and uncertain information. Consequently we may have programs where various degrees of contradictory information can be concluded. In order to define a semantics with these requirements, we generalize the paraconsistent well-founded semantics for extended logic programs, ($WFSX_p$) [2]. As we do not impose any specific characterization for explicit negation, we can introduce in our framework any negation operator supported by Fitting's bilattice. In the sequence, we also present a semantics based on Coherent Answer Sets.

The paper is structured as follows: in section 2, we present Fitting's bilattices [8] and we use this framework to characterize the default negation operator. Section 3 presents the core of our work: there we introduce Paraconsistent Logic Programs and their semantics. In Section 4, we show a illustrative example where our framework is used to encode a rather complex decision table. Afterwards, we present the embedding of $WFSX_p$ into our semantics. Finally, we draw some conclusions and mention future work.

2 Bilattices

Aiming at the characterization of uncertainty, several varieties of fixed point semantics for logic programming have been proposed. In common, they require a suitable machinery to ensure the existence of fixed points of the operator associated with a program. For the machinery to work smoothly, there must be an appropriate interaction between the logical operations allowed in the programming language and the underlying partial ordering of truth values. It is claimed by Fitting in [8] that the notion of bilattice, as presented by Ginsberg [12], can be applied to most kinds of fixed point semantics and provides an account of the intended partial ordering, for each of the truth functional connectives, and the interactions between them. Following Fitting's ideas, we recall below how to construct a bilattice:

Definition 1 (Bilattice construction). *Given two complete lattices $\langle C, \leq_1 \rangle$ and $\langle D, \leq_2 \rangle$ the structure $\mathcal{B}(C, D) = \langle C \times D, \leq_k, \leq_t \rangle$ is a complete bilattice, where the partial orderings are defined as follows:*

$$\langle c_1, d_1 \rangle \leq_k \langle c_2, d_2 \rangle \text{ if } c_1 \leq_1 c_2 \text{ and } d_1 \leq_2 d_2$$
$$\langle c_1, d_1 \rangle \leq_t \langle c_2, d_2 \rangle \text{ if } c_1 \leq_1 c_2 \text{ and } d_2 \leq_2 d_1$$

To each ordering are associated join ($\oplus$) and meet ($\otimes$) operations according to the following equations:

Knowledge ordering ($\leq_k$)	*Truth ordering* ($\leq_t$)
$\langle c_1, d_1 \rangle \otimes_k \langle c_2, d_2 \rangle = \langle c_1 \sqcap_1 c_2, d_1 \sqcap_2 d_2 \rangle$	$\langle c_1, d_1 \rangle \otimes_t \langle c_2, d_2 \rangle = \langle c_1 \sqcap_1 c_2, d_1 \sqcup_2 d_2 \rangle$
$\langle c_1, d_1 \rangle \oplus_k \langle c_2, d_2 \rangle = \langle c_1 \sqcup_1 c_2, d_1 \sqcup_2 d_2 \rangle$	$\langle c_1, d_1 \rangle \oplus_t \langle c_2, d_2 \rangle = \langle c_1 \sqcup_1 c_2, d_1 \sqcap_2 d_2 \rangle$

The intuition here is that C provides the evidence for (and D evidence against or doubt) believing in the truth of a statement. If $\langle c_1, d_1 \rangle \leq_k \langle c_2, d_2 \rangle$, then situation 2 has more information than 1, i.e. knowledge is increased, both in truth and in falsity. If $\langle c_1, d_1 \rangle \leq_t \langle c_2, d_2 \rangle$, then we have more reasons to believe in situation 2 than in 1, because the reasons for believing the statement either increased or the reasons against it are weaker, i.e. $\langle c_2, d_2 \rangle$ is truer than $\langle c_1, d_1 \rangle$.

Observe that negation is not granted by this construction. In order to introduce it, Ginsberg [12] requires that the negation mesh well with the orderings and satisfy the double negation property:

Definition 2 (Negation). *A bilattice $\mathcal{B}(C, D)$ has a negation operation if there is a mapping $\neg : C \times D \to C \times D$ such that*

1. $a \leq_k b \Rightarrow \neg a \leq_k \neg b$;
2. $a \leq_t b \Rightarrow \neg b \leq_t \neg a$;
3. $\neg\neg a = a$.

Fitting [8] introduced one more basic operator on bilattices: conflation (see [8] for more details).

Definition 3 (Conflation). *A bilattice $\mathcal{B}(C, D)$ enjoys a conflation operation if there is a mapping $- : C \times D \to C \times D$ such that*

1. $a \leq_k b \Rightarrow -b \leq_k -a$;
2. $a \leq_t b \Rightarrow -a \leq_t -b$;
3. $- - a = a$.

Hence, negation should reverse the degree of truth, but preserve that of knowledge. Conflation is similar, but with the roles of $\leq_k$ and $\leq_t$ exchanged. Moreover, if $\neg\neg a = a$ (resp. $- - a = a$) is not verified, the negation (resp. conflation) operator is said to be weak.

In [8], Fitting further states the conflation operator results as moving to "default" evidence. This means that given $L \in C \times D$, where $\mathcal{B}(C, D)$ is a bilattice with the conflation operator, in $-L$ we are to count as "for" whatever did not count as "against" before, and "against" what did not count as "for". Thus, $-\neg L$ resembles *not* L, where *not* is the default negation operator found in many logic programming formalisms. Similarly $-\neg L$ also resembles $\mathbf{B}\neg L$, where $\mathbf{B}$ is the belief operator of Auto-Epistemic logic of Belief (AEB) [1]. Based on this pondering, we define here the default negation operator:

Definition 4 (Default Negation). *Let $\mathcal{B}(C, D)$ a bilattice. Consider $\neg$ and $-$ respectively a negation and a conflation operator on $\mathcal{B}(C, D)$. We define not $: C \times D \to C \times D$ as the default negation operator where*

$$\textit{not}\ L = -\neg L$$

The default operator plays a central role in our framework. In Section 3, following ideas expressed in [20], we will relate it with the explicit negation operator through the *coherence principle*.

3 Paraconsistent Logic Programs

In the framework of Monotonic Logic Programs [4], the rules are constituted by arbitrary isotonic[1] body functions and by propositional symbols in the head. In [3], Damásio and Pereira extend the syntax of Monotonic Logic Programs allowing for rules with antitonic bodies (Antitonic Logic Programs), of which normal logic programs are a special case. The authors also show that all the standard logic programming theoretical results carry over to such Antitonic Logic Programs through both Stable and Well-founded semantics alike. Because of their arbitrary monotonic and antitonic operators over a complete lattice, these programs pave the way to combine and integrate into a single framework several forms of reasoning, such as fuzzy, probabilistic, uncertain, and paraconsistent ones.

Many works (e.g. [11, 14, 25, 20]) have argued about the convenience of introducing into logic programming a way to distinguish what can be shown to be false from what is false by default because it cannot be proven true. So called Extended Logic Programs add explicit negation to normal programs. In [20], it is claimed that explicit negation should entail default negation. This is the *coherence principle*. Unfortunately antitonic logic programs are not suitable to characterize this principle as we can see below.

[1] A function is isotonic (antitonic) iff the value of the function increases (decreases) when we increase any argument while the remaining arguments are kept fixed.

Following another path, Lakshmanan and Sadri [15] proposed a framework for modeling uncertainty, where both belief and doubt are explicitly incorporated. Motivated by these two formalisms, extended logic programs and Lakshmanan and Sadri's framework, we shall extend the syntax of antitonic logic programs to an arbitrary complete bilattice of truth-values. Then we employ the characterization of the default negation operator presented in the section 2 to impose coherence on it.

With relation to semantics, we follow a paraconsistent approach based on $WFSX_p$ [2], one of the most important semantics for extended logic programs. When defining paraconsistent logic programs, we adopt an algebraic characterization of both the language and interpretations of operators. This is a very general and powerful framework, allowing for a simple relation between the syntax and its semantics. We start its presentation by introducing the syntax of paraconsistent logic programs:

Definition 5 ($\mathcal{B}$-algebras). *Consider the complete bilattice $\mathcal{B}(C, C)$ with negation ($\neg$) and conflation ($-$) operators, where C is a given complete lattice. We define the $\mathcal{B}$-algebra $\mathfrak{P}$ associated to $\mathcal{B}(C, C)$ as an algebra whose carrier set is $C \times C$, and whose operators must include at least $\neg, -$, the operator $\otimes_k$ of $\mathcal{B}(C, C)$, and constants corresponding to the elements $< c, d >$ in $C \times C$.*

Having defined the alphabet, we can present the corresponding algebra of expressions, frow which we select the subalgebra of well-formed formulae:

Definition 6 (Formulae). *Let $\mathfrak{P}$ be the $\mathcal{B}$-algebra associated with a complete bilattice. We define the alphabet $A_{\Pi,\Delta}$ as the disjoint union $\Pi \cup \Delta \cup S$, where Π is a countably infinite set of propositional symbols, Δ is the set of operators found in $\mathfrak{P}$, and S is the set of auxiliary symbols "(", ")", and ",". Furthermore, consider the algebra $\mathfrak{E}$ of expressions corresponding to the alphabet $A_{\Pi,\Delta}$. As usual, we define the well-formed formulae (in short formulae) generated by Δ over Π as the elements of the carrier set of the least subalgebra of $\mathfrak{E}$ containing Π. We named such subalgebra of $FORM_{\mathfrak{P}}(\Pi)$.*

Given the definitions above, Paraconsistent Logic Programs are defined as follows:

Definition 7 (Paraconsistent Logic Programs). *Let $\mathfrak{P}$ be the $\mathcal{B}$-algebra associated with a complete bilattice $\mathcal{B}(C, C)$, Π a set of propositional symbols, and $FORM_{\mathfrak{P}}(\Pi)$ the corresponding algebra of freely generated formulae. A paraconsistent logic program P is a set of rules of the form $A \leftarrow \Psi$ such that:*

1. *The body formula Ψ is any formula of $FORM_{\mathfrak{P}}(\Pi)$;*
2. *The head A of the rule is a propositional symbol of Π;*
3. *Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ $(m, n \geq 0)$ be a sequence with all occurrences of propositional symbols in Ψ. If $A \leftarrow \Psi$ belongs to P, then Ψ corresponds to a function having $A_1, \ldots, A_m, B_1, \ldots, B_n$ as arguments and which is isotonic (antitonic) with relation to $A_1, \ldots, A_m$ (resp. $B_1, \ldots, B_n$). Sometimes we will denote a rule $A \leftarrow \Psi$ of P by $A \leftarrow \Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n]$.*

We assume, in the rest of this work, a $\mathcal{B}$-algebra $\mathfrak{P}$ associated to a complete bilattice $\mathcal{B}(C,C)$. We will add the subscript $\mathfrak{P}$ to the "symbols" of the operators to denote they are defined in $\mathfrak{P}$. However, when may cause no confusion, we will deliberately omit this subscript from the operators. We also assume the set with the constants corresponding to the elements in $C \times C$ is implicit in the $\mathcal{B}$-algebra and we will do any reference to it in the examples. Consider still that a set Π of propositional symbols and the corresponding algebra $FORM_{\mathfrak{P}}(\Pi)$ are given.

We refer to P as monotonic logic program if for each rule $A \leftarrow \Psi$ in P, the body formula Ψ corresponds to a isotonic function, i.e. rules are of the kind $A \leftarrow \Psi[A_1, \ldots, A_m \mid\ ]$. We refer to P as antitonic logic program if each rule is either of the kind $A \leftarrow \Psi[A_1, \ldots, A_m \mid\ ]$ or $A \leftarrow \Psi[\ \mid B_1, \ldots, B_n]$. Mark the body formula Ψ can be isotonic with relation to some occurrences of a propositional symbol A and antitonic with relation to other occurrences of the same propositional symbol A. In order to illustrate this we proffer the example below:

Example 1. Let $\mathcal{B}([0,1],[0,1])$ be a bilattice and $\mathfrak{P} = \langle [0,1] \times [0,1], \otimes_k, \neg, - \rangle$ be an $\mathcal{B}$-algebra where $\otimes_k(\langle \alpha, \beta \rangle, \langle \gamma, \delta \rangle)$[2] $= \langle min(\alpha,\gamma), min(\beta,\delta) \rangle$[3], $\neg(\langle \alpha, \beta \rangle) = \langle \beta, \alpha \rangle$, and $-(\langle \alpha, \beta \rangle) = \langle 1-\beta, 1-\alpha \rangle$. Here is a paraconsistent logic program:

$$
\begin{array}{lll}
c \leftarrow \neg b \ \otimes_k \ -b. & a \leftarrow \langle 0.8, 0.6 \rangle . & e \leftarrow - \neg d. \\
b \leftarrow \neg\neg a \ \otimes_k \ \langle 0.9, 1 \rangle . & d \leftarrow -c. & e \leftarrow \neg d.
\end{array}
$$

In the above program, the only rule for the propositional symbol c ($c \leftarrow \neg b \,\&\, -b$) has a body corresponding to a function which is isotonic with relation to the first occurrence of b (in $\neg b$) and is antitonic with relation to the latter occurrence of b (in $-b$).

We proceed by exhibiting the semantics for paraconsistent logic programs. As usual, an interpretation is simply an assignment of truth-value to each propositional symbol in the language.

Definition 8 (Interpretation). *An interpretation is a mapping $I : \Pi \to C \times C$. By the unique homomorphic extension theorem, the interpretation extends uniquely to a valuation $\hat{I} : FORM_{\mathfrak{P}}(\Pi) \to C \times C$. The set of all interpretations of the formulas with respect to the algebra $\mathfrak{P}$ is denoted by $\mathcal{I}_{\mathfrak{P}}$.*

The unique homomorphic extension theorem guarantees that for every interpretation of propositional symbols there is an unique associated valuation function. The ordering $\leq_k$ on the truth-values in $\mathfrak{P}$ is extended to the set of interpretations as follows:

Definition 9 (Lattice of Interpretations). *Consider $\mathcal{I}_{\mathfrak{P}}$ the set of all interpretations with respect to algebra $\mathfrak{P}$, and two interpretations $I_1, I_2 \in \mathcal{I}_{\mathfrak{P}}$. Then, $< \mathcal{I}_{\mathfrak{P}}, \sqsubseteq >$ is a complete lattice where $I_1 \sqsubseteq I_2$ iff $\forall_{p \in \Pi}\ I_1(p) \leq_k I_2(p)$. The least*

[2] Using infix notation we have $\langle \alpha, \beta \rangle \otimes_k \langle \gamma, \delta \rangle$.

[3] The function *min* returns the minimum of two real numbers.

interpretation Δ *maps every propositional symbol to the least element under* $\leq_k$ *of* $C \times C$*, and the greatest interpretation* ∇ *maps every propositional symbol to the top element under* $\leq_k$ *of the complete bilattice of truth-values* $C \times C$*.*

Extending the results of van Emden and Kowalski [23] to the general theoretical setting of algebras, we define below the immediate consequences operator:

Definition 10 (Immediate Consequences Operator). *Let* P *be a monotonic logic program. Define the immediate consequences operator* $T_P^{\mathfrak{P}} : \mathcal{I}_{\mathfrak{P}} \to \mathcal{I}_{\mathfrak{P}}$*, mapping interpretations to interpretations:*

$$T_P^{\mathfrak{P}}(I)(A) = lub_k \left\{ \hat{I}(\Psi) \text{ such that } A \leftarrow \Psi \in P \right\},$$

where A *is a propositional symbol, and* lub_k *is the least upper bound under knowledge ordering.*

The use of the knowledge ordering is fundamental. Assume that we state that a proposition, say a, is both true and false via the rules $a \leftarrow < \top, \bot >$ and $a \leftarrow < \bot, \top >$. Using the truth-ordering, we would obtain in the least model that a is true, whilst under knowledge ordering a is mapped to $< \top, \top >$. For capturing default negation, a new notion of interpretation is also required:

Definition 11 (Partial Interpretations). *A partial interpretation is a pair of interpretations* $< I^t, I^{tu} >$*. The set of all partial interpretations is* $\mathcal{I}_{\mathfrak{P}}^p$*.*

The I^t component contains what is "true" in the interpretation, while I^{tu} contains what is "non-false", i.e. true or undefined. It is important to mark that consistency of interpretations (i.e. $I^t \sqsubseteq I^{tu}$) is not imposed and, consequently, paraconsistency is allowed, i.e. something may be "true and false" (by being not "true or undefined"). Two orders among partial interpretations are useful:

Definition 12 (Standard and Fitting Ordering). *Let* I_1 *and* I_2 *be two partial interpretations. The standard and Fitting orderings among partial interpretations are defined by:*

Standard ordering: $I_1 \sqsubseteq_s I_2$ *iff* $I_1^t \sqsubseteq I_2^t$ *and* $I_1^{tu} \sqsubseteq I_2^{tu}$.
Fitting ordering: $I_1 \sqsubseteq_f I_2$ *iff* $I_1^t \sqsubseteq I_2^t$ *and* $I_2^{tu} \sqsubseteq I_1^{tu}$.

The standard ordering prefers partial interpretations with propositional symbols having higher true truth-value and lesser false truth-value. The Fitting ordering, prefers interpretations with less undefinedness.

The set of partial interpretations ordered by $\sqsubseteq_s$ *or* $\sqsubseteq_f$ *is a complete lattice. Clearly, the bottom and top elements of these lattice are* $\bot_s =< \Delta, \Delta >$ *(viz. all is false),* $\top_s =< \nabla, \nabla >$ *(viz. all is true),* $\bot_f =< \Delta, \nabla >$ *(viz. all is undefined) and* $\top_f =< \nabla, \Delta >$ *(viz. all is true and false).*

Given a partial interpretation, the model of a paraconsistent logic program is defined as per below:

Definition 13 (Models). *Let P be a paraconsistent logic program and $I = < I^t, I^{tu} >$ a partial interpretation. Consider a formula $\Psi[A_1, \dots A_m \mid B_1, \dots, B_n]$ in $FORM_{\mathfrak{P}}(\Pi)$. We define $\hat{I}(\Psi) = \Psi_{\mathfrak{P}}[I^t(A_1), \dots I^t(A_m) \mid I^{tu}(B_1), \dots, I^{tu}(B_n)]$. Note this is an extension of the notion of interpretation (definition 8) to the case of partial interpretation. Furthermore, the partial interpretation I satisfies a rule $A \leftarrow \Psi$ of P iff:*

$$\hat{I}(\Psi) \leq_k I(A)$$

We say that I is a model of P iff I satisfies all rules of P.

The next definition is based on the usual way of defining well-founded and stable model semantics for normal logic programs. It is a Gelfond-Lifschitz like division operator [10] which transforms Paraconsistent Logic Programs into Monotonic ones, for which we know how to compute the corresponding least models through the immediate consequences operator.

Definition 14 (Program Division). *Consider a paraconsistent logic program P and an interpretation*[4] *I. The division of program P by I is the monotonic logic program:*

$$\frac{P}{I} = \Big\{A \leftarrow \Psi[A_1, \dots, A_m \mid I(B_1), \dots I(B_n)] \quad \textit{such that } A \leftarrow \Psi[A_1, \dots, A_m \mid B_1, \dots B_n] \in P\Big\}$$

The program division $\frac{P}{I}$ evaluates according to I all occurrences of antitonic propositional symbols in P. Then the least model of the resulting program can be determined by resorting to a Γ operator:

Definition 15 (Gamma operator). *Let P be a paraconsistent logic program and J an interpretation. Define*

$$\Gamma_P^{\mathfrak{P}}(J) = \text{lfp } T_{\frac{P}{J}}^{\mathfrak{P}} = T_{\frac{P}{J}}^{\mathfrak{P}}\uparrow^{\lambda}, \textit{ for some ordinal } \lambda$$

Notwithstanding, according to [20], negation and default negation are not unrelated: the coherence principle ensures that the former entails the latter. From an epistemic viewpoint, coherence can be seen as an instance of the necessitation principle, which states that if something is known then it is believed. Therefore, one must have $A \leq_k \textit{not } \neg A$ (coherence principle). As $\textit{not } \neg A = -A$ (definition 4), we have to guarantee $A \leq_k -A$. Given these ponderations, in order to enforce coherence, we will resort to the semi-normal program, which is a generalization of the approach taken in the definition of the paraconsistent well-founded semantics with explicit negation $WFSX_p$ [2]:

Definition 16 (Semi-normal program). *Let P be a paraconsistent logic program. The semi-normal version of P, denoted P_s, is the paraconsistent logic program obtained from P replacing every $A \leftarrow \Psi$ belonging to P by $A \leftarrow \Psi \otimes_k -A$.*

[4] Mark well this is an interpretation, not a partial interpretation!

Notice that as Antitonic Logic Programs do not support semi-normal programs, they can not characterize adequately the coherence principle. In [2], an alternating fixpoint definition of $WFSX_p$ is provided, relying on the application of two anti-monotonic operators, $\Gamma_P \Gamma_{P_s}$. Before generalizing this result to our framework we need to guarantee the following theorem:

Theorem 1. *Consider the paraconsistent logic program P. Let I_1 and I_2 be two interpretations such that $I_1 \leq I_2$. Then $\Gamma_P^{\mathfrak{P}} \Gamma_{P_s}^{\mathfrak{P}}(I_1) \leq \Gamma_P^{\mathfrak{P}} \Gamma_{P_s}^{\mathfrak{P}}(I_2)$.*

Thus we can define the paraconsistent well-founded semantics:

Definition 17 (Paraconsistent Well-founded Semantics). *Consider a paraconsistent logic program P and a partial interpretation $M = \langle M^t, M^{tu} \rangle$. We say M is a partial paraconsistent model for P iff $M^t = \Gamma_P^{\mathfrak{P}}(\Gamma_{P_s}^{\mathfrak{P}}(M^t))$ and $M^{tu} = \Gamma_{P_s}^{\mathfrak{P}}(M^t)$.*

The least partial paraconsistent model under the Fitting ordering is defined as the paraconsistent well-founded model $WFM_p(P)$, and can be obtained by iterating $\Gamma_P^{\mathfrak{P}} \Gamma_{P_s}^{\mathfrak{P}}$ from Δ. Then, given that I_α is the least fixpoint of $\Gamma_P^{\mathfrak{P}} \Gamma_{P_s}^{\mathfrak{P}}$, we shall have $WFM_p(P) = \left\langle \Gamma_P^{\mathfrak{P}}(\Gamma_{P_s}^{\mathfrak{P}}(I_\alpha)), \Gamma_{P_s}^{\mathfrak{P}}(I_\alpha) \right\rangle$.

Given that $M = \langle M^t, M^{tu} \rangle$ is a partial stable model, we say an atom A is:

- **true** *with degree ϑ wrt. M if $\vartheta \leq_k M^t(A)$ and $\vartheta \leq_k M^{tu}(A)$.*
- **undefined** *with degree ϑ wrt. M if $\vartheta \not\leq_k M^t(A)$ and $\vartheta \leq_k M^{tu}(A)$.*
- **false** *with degree ϑ wrt. M if $\vartheta \not\leq_k M^t(A)$ and $\vartheta \not\leq_k M^{tu}(A)$.*
- **inconsistent** *with degree ϑ wrt. M if $\vartheta \leq_k M^t(A)$ and $\vartheta \not\leq_k M^{tu}(A)$.*

Letting $\Omega_P^{\mathfrak{P}}(J) = \langle \Gamma_P^{\mathfrak{P}}(J^{tu}), \Gamma_{P_s}^{\mathfrak{P}}(J^t) \rangle$, the paraconsistent partial stable models of the program P can be seen as the fixpoints of $\Omega_P^{\mathfrak{P}}$. Since $\Omega_P^{\mathfrak{P}}$ is monotonic with respect to $\sqsubseteq_f$, the least fixpoint under Fitting ordering (the paraconsistent well-founded model) is guaranteed to exist. Notice that $\Omega_P^{\mathfrak{P}}$ is not monotonic with respect to $\sqsubseteq_s$, thus it cannot be ensured the existence of the least fixpoint under the Standard ordering. This justifies why the Fitting ordering is used to define WFM_p. Next we define the coherent answer sets:

Definition 18 (Coherent Answer Sets). *Let P be a paraconsistent logic program. A coherent answer set is a partial paraconsistent model of the form $< M, M >$.*

As usual, coherent answer sets are not guaranteed to exist. However, as for paraconsistent well-founded models, coherent answer sets do comply with the coherence principle:

Proposition 1. *Let P be a paraconsistent logic program. Then all partial paraconsistent models $M = \langle M^t, M^{tu} \rangle$ for P obeys the coherence principle. This requires that $\hat{M}(\neg L) \sqsubseteq \hat{M}(-\neg L)$.*

Consequently, all coherent answer sets and the paraconsistent well-founded model for P observe the coherence principle.

In the next section, we show an example motivating the applicability of our framework and of its two proposed semantics.

4 Example

Paraconsistent Logic Programs have a large range of applications. We will use the following example adapted from [19] to encode a rather complex decision table based on rough relations. Such relations are determined from *Rough Sets* (cf. [19]), introduced to deal with imprecise information. In [17], a logic programming language, *Rough Datalog*, is defined, making possible to describe systems of rough relations and to reason about them. On some points, Rough Datalog resembles our framework. However, Paraconsistent Logic Programs appear to be more generic.

The symptoms *fever*, *cough*, *headache*, and *muscle-pain* are used to decide whether a patient has a flu. The diagnosis is performed according to the decision table[5] in Table 1:

Table 1. Decision table for flu

fever	*cough*	*headache*	*muscle-pain*	*flu*
no	no	no	no	no (in 99% of the cases)
yes	no	no	no	no (in 80% of the cases)
yes	yes	no	no	no (in 30% of the cases)
yes	yes	no	no	yes (in 60% of the cases)
yes	yes	yes	yes	yes (in 75% of the cases)

Mark that in the 3rd and 4th lines of table 1 we have contradictory conclusions for the same set of symptoms, and that in 10% of the cases the physician remains undecided.

We resort to the bilattice $\mathcal{B}([0,1],[0,1])$ to encode the above decision table. We also assume an $\mathcal{B}$-algebra $\mathfrak{P}$, whose carrier set is [0,1] $\times[0,1]$. The $\mathcal{B}$-algebra $\mathfrak{P}$ has the operators $\neg(\langle\alpha,\beta\rangle) = \langle\beta,\alpha\rangle$, $-(\langle\alpha,\beta\rangle) = \langle 1-\beta, 1-\alpha\rangle$, and $\otimes_k(\langle\alpha,\beta\rangle, \langle\gamma,\delta\rangle) = \langle min(\alpha,\gamma), min(\beta,\delta)\rangle$. The first case can be represented by

$$\neg flu \leftarrow (< 0.99, 0.0 > \otimes_k \neg fever \otimes_k \neg cough \otimes_k \neg headache \otimes_k \neg muscle\_pain)$$

Such rule can not be supported by the language of our framework because it only admit rules whose heads are atoms. However, as the reader can check, the first rule can be alternatively written as

$$flu \leftarrow \neg\,(< 0.99, 0.0 > \otimes_k \neg fever \otimes_k \neg cough \otimes_k \neg headache \otimes_k \neg muscle\_pain)$$

which reduces to[6]

$$flu \leftarrow < 0.0, 0.99 > \otimes_k fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain \qquad (1)$$

[5] The figures are fictitious.

[6] From [8], we have $\neg(A \otimes_k B) = (\neg A \otimes_k \neg B)$.

Similarly, the second diagnosis case can be implemented via the following rule:

$$flu \leftarrow < 0.0, 0.80 > \otimes_k \neg fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain$$

The translation of the last case is immediate:

$$flu \leftarrow < 0.75, 0.0 > \otimes_k fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain \qquad (2)$$

The reader surely will notice that the body of rule 2 is almost identical to that of rule 1, with the exception of the confidence degrees in the rules. According to WFM_p, the truth-value of flu is determined by the least upper bound (under $\leq_k$) of the truth-values assigned to the bodies of the rules for flu. Consequently, the rules 1 and 2 can be combined into a single one:

$$flu \leftarrow < 0.75, 0.99 > \otimes_k fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain \qquad (3)$$

Thus, the above rule expresses both positive and negative evidence for diagnosing a flu:

- if a patient has fever, cough, headache, and muscle-pain, then flu is a correct diagnosis in 75% of the cases.
- if a patient **doesn't** have fever, **doesn't** cough, **doesn't** have neither headache nor muscle-pain, then he **doesn't** have flu in 99% of the situations.

So, the positive evidence for the consequent is only concluded when all the propositions in the body of the rule have positive evidence for them. Symmetrically, the negative evidence for the conclusion is only gotten when all propositions in the body have negative evidence.

For the remaining situation (fever,cough,not headache, and not muscle_pain) two distinct rules are required for concluding both that the patient might have or not have a flu:

$$flu \leftarrow < 0.0, 0.3 > \otimes_k \neg fever \otimes_k \neg cough \otimes_k headache \otimes_k muscle\_pain$$

$$flu \leftarrow < 0.60, 0.0 > \otimes_k fever \otimes_k cough \otimes_k \neg headache \otimes_k \neg muscle\_pain$$

Assume now that antibiotics are prescribed when flu is not concluded. We will compare two possible translations of this statement, represented by either of the following rules:

$$antibiotics \leftarrow \neg flu \qquad (4)$$

or

$$antibiotics \leftarrow -\neg flu \qquad (5)$$

The former allows us to conclude that antibiotics should be prescribed when there is explicit negative evidence for flu. With the latter rule, antibiotics are recommended when there is no evidence[7] for flu. The rules for diagnosing flu are

[7] This is not the same has having negative evidence!

$$
\begin{array}{l}
flu \leftarrow < 0.0, 0.80 > \ \otimes_k \neg fever \otimes_k \ \ cough \otimes_k \ \ headache \otimes_k \ \ muscle\_pain \\
flu \leftarrow < 0.75, 0.99 > \otimes_k \ \ fever \otimes_k \ \ cough \otimes_k \ \ headache \otimes_k \ \ muscle\_pain \\
flu \leftarrow \ < 0.0, 0.3 > \ \otimes_k \neg fever \otimes_k \neg cough \otimes_k \ \ headache \otimes_k \ \ muscle\_pain \\
flu \leftarrow < 0.60, 0.0 > \ \otimes_k \ \ fever \otimes_k \ \ cough \otimes_k \neg headache \otimes_k \neg muscle\_pain
\end{array}
$$

We now illustrate the behaviour of paraconsistent well-founded semantics in several situations. Table 2 contains six different models of the above program. The first line of every model corresponds to the T component of the model, while the second line represents the TU component of the same model. The five leftmost columns represent the interpretation, and the two rightmost columns the confidence degrees of explicit and of default negations.

Table 2. Models for the flu program

fever	cough	headache	muscle-pain	flu	¬flu	−¬flu
$<0,1>$	$<0,1>$	$<0,1>$	$<0,1>$	$<0,0.99>$	$<0.99,0>$	$<1,0.01>$
$<0,1>$	$<0,1>$	$<0,1>$	$<0,1>$	$<0,0.99>$	$<0.99,0>$	$<1,0.01>$
$<1,0>$	$<0,1>$	$<0,1>$	$<0,1>$	$<0,0.8>$	$<0.8,0>$	$<1,0.2>$
$<1,0>$	$<0,1>$	$<0,1>$	$<0,1>$	$<0,0.8>$	$<0.8,0>$	$<1,0.2>$
$<1,0>$	$<1,0>$	$<0,1>$	$<0,1>$	$<0.6,0.3>$	$<0.3,0.6>$	$<0.4,0.7>$
$<1,0>$	$<1,0>$	$<0,1>$	$<0,1>$	$<0.6,0.3>$	$<0.3,0.6>$	$<0.4,0.7>$
$<1,0>$	$<1,0>$	$<1,0>$	$<1,0>$	$<0.75,0>$	$<0,0.75>$	$<0.25,1>$
$<1,0>$	$<1,0>$	$<1,0>$	$<1,0>$	$<0.75,0>$	$<0,0.75>$	$<0.25,1>$
$<0.4,0.6>$	$<0.7,0.3>$	$<0.1,0.9>$	$<0.2,0.7>$	$<0.4,0.3>$	$<0.3,0.4>$	$<0.6,0.7>$
$<0.4,0.6>$	$<0.7,0.3>$	$<0.1,0.9>$	$<0.2,0.7>$	$<0.4,0.3>$	$<0.3,0.4>$	$<0.6,0.7>$
$<0.4,0.6>$	$<0.7,0.3>$	$<0.7,0.9>$	$<0.2,0.7>$	$<0.4,0.3>$	$<0.3,0.4>$	$<0.7,0.7>$
$<0.4,0.6>$	$<0.7,0.3>$	$<0.1,0.3>$	$<0.2,0.7>$	$<0.3,0.3>$	$<0.3,0.3>$	$<0.6,0.7>$

The first model in Table 2 is obtained by adding the following set of facts (the confidence degrees are extracted from the T component of the model) to the previous rules :

$$fever \leftarrow <0,1> \quad cough \leftarrow <0,1> \quad headache \leftarrow <0,1> \quad muscle\_pain \leftarrow <0,1>$$

The first four models correspond to the 4 cases about diagnosing flu. As the reader can easily check, the column for flu is in accordance with the evidence expressed in Table 1.

The distinctive effect of the rules $antibiotics \leftarrow \neg flu$ and $antibiotics \leftarrow -\neg flu$ can be observed from the columns for $\neg flu$ and $-\neg flu$. By the coherence principle, it is always the case that for every model $\hat{M}(\neg flu) \leq_k \hat{M}(-\neg flu)$ as the reader can check in the T-lines of the models. For instance, in the second model we derive $\neg flu$ with degree of evidence $< 0.8, 0 >$ and $-\neg flu$ with evidence $< 1, 0.2 >$, and as expected $< 0.8, 0 > \leq_k < 1, 0.2 >$. Since flu has value $< 0, 0.8 >$, and $-\neg flu$ has value $< 1, 0.2 >$, antibiotics should be prescribed according to rule 5 as there is no positive evidence for having a flu. The situation where flu is diagnosed appears in the fourth model, and so antibiotics are not prescribed.

In the fifth model the physician is not certain regarding all symptoms. The interesting aspect of this case, is that the degree of evidence for flu is obtained by combining the degrees of evidence of several rules. Finally, the last model illustrates how paraconsistency is handled by our semantics. The fact $headache \leftarrow < 0.7, 0.9 >$ is inconsistent in Fitting's sense (and thus also inconsistent in the WFM_p). Since flu is true with degree $< 0.4, 0.3 >$ and true or undefined with degree $< 0.3, 0.3 >$ then flu is inconsistent in the well-founded model of the program. All other models are consistent and fully-defined, i.e. they are coherent answer sets.

5 Embedding of $\boldsymbol{WFSX_p}$

Extended Logic Programs are defined as a set of rules of the form

$$A \leftarrow B_1, \ldots, B_m, not\ C_1, \ldots, not\ C_n,$$

where A, B_i, and C_j $(1 \leq i \leq m, 1 \leq j \leq n)$ are atoms or the explicit negation of atoms. Since these programs do not admit rules with confidence degrees, if we decide to encode the decision table in Table 1 (see previous section), we will conclude a patient has (or does not have) flu in 100% of the cases:

Example 2. The (crisp) rules for diagnosing flu are represented by the following extended logic program:

$$\begin{array}{r}
\neg flu \leftarrow \neg fever, \neg cough, \neg headache, \neg muscle_pain \\
\neg flu \leftarrow fever\ ,\ cough, \neg headache, \neg muscle_pain \\
\neg flu \leftarrow fever\ ,\ cough, \neg headache, \neg muscle_pain \\
flu \leftarrow fever\ ,\ cough, \neg headache, \neg muscle_pain \\
flu \leftarrow fever\ ,\ cough,\ headache,\ muscle_pain
\end{array}$$

It is clear extended logic programs can be easily embedded into our framework:

Definition 19. *Let P be an extended logic program. Consider the algebra $\mathfrak{P}$ with respect to the bilattice $\mathcal{B}(\{0,1\},\{0,1\})$ with the operators $-$ and $\neg$, where $\neg(\langle a, b\rangle) = \langle b, a\rangle$ and $-(\langle a, b\rangle) = \langle 1-b, 1-a\rangle$. Define P_w as a paraconsistent logic program over $\mathfrak{P}$ such that*

- *For each rule* $A \leftarrow B_1, \ldots B_m, not\ C_1, \ldots, not\ C_n$ *(*$m, n \geq 0$*) belonging to* P*, we have* $A \leftarrow [1, 0] \otimes_k B_1 \otimes_k \ldots \otimes_k B_m \otimes_k -\neg C_1 \otimes_k \ldots \otimes_k -\neg C_n$ *in* P_w*;*
- *For each rule* $\neg A \leftarrow B_1, \ldots B_m, not\ C_1, \ldots, not\ C_n$ *(*$m, n \geq 0$*) belonging to* P*, we have* $A \leftarrow [0, 1] \otimes_k \neg B_1 \otimes_k \ldots \otimes_k \neg B_m \otimes_k -C_1 \otimes_k \ldots \otimes_k -C_n$ *in* P_w*.*

Moreover, let I *and* I_w *be respectively interpretations in* $WFSX_p$ *and* WFM_p *senses. We say* I *is a translation of* I_w *(and vice-versa), denoted by* $I \cong I_w$*, iff for each atom* A*,* $I_w(A) = \langle 1, 1 \rangle$ *iff* $\{A, \neg A\} \subseteq I$*;* $I_w(A) = \langle 1, 0 \rangle$ *iff* $A \in I$ *and* $\neg A \notin I$*;* $I_w(A) = \langle 0, 1 \rangle$ *iff* $A \notin I$ *and* $\neg A \in I$*;* $I_w(A) = \langle 0, 0 \rangle$ *iff* $A \notin I$ *and* $\neg A \notin I$*.*

Note the program P_w is syntactically very similar to the program of the previous section, except confidence degrees can be only $\langle 1, 0 \rangle$ for positive evidences and $\langle 0, 1 \rangle$ for negative ones. Given the above definition, the embedding of the semantics is straightforward:

Theorem 2. *Let* P *be an extended logic program with well-founded model* $T \cup not\ F$ *, and* P_w *the corresponding paraconsistent logic program with the model* $WFM_p(P_w) = \langle M^t, M^{tu} \rangle$*. Then we have* $T \cong M^t$ *and* $\Gamma_s T \cong M^{tu}$*.*

Thus our framework captures $WFSX_p$, one of the main paraconsistent semantics for Extended Logic Programs. Following the results for $WFSX_p$ exhibited in [2], we can state that if we use $\mathcal{B}(\{0, 1\}, \{0, 1\})$ as bilattice of truth-values, the definition 18 of coherent answers coincides with the definition of answer sets presented in [11]. Furthermore, we can see Antitonic Logic Programs as a type of Paraconsistent Logic Program where each body formula either corresponds to a monotonic or to an antitonic function, and where doubt in each propositional symbol of Π is mapped into the bottom element of the underlying ordering of the associated bilattice. Since we generalize Antitonic Logic Programs (assuming conflation and negation operators exist), we easily extend all the semantics for which we have an embedding into these programs, such as Probabilistic Logic Programs [15], Hybrid Probabilistic Logic Programs [5], Possibilistic Logic Programming [6], and Fuzzy Logic Programs [24].

6 Conclusion

We have defined Paraconsistent Logic Programs, which are a generalization of Antitonic Logic Programs [3]. An important merit of this work is the introduction into a rather general framework, of an appropriate kind, of the concepts that cope with explicit negation and default negation. It is certified that default negation complies with the coherence principle. Program rules have bodies corresponding to compositions of arbitrary monotonic and antitonic operators over a complete bilattice in the sense of Fitting [8], and provide an elegant way to present belief and doubt. In order to define its semantics we resort, as usual, to a program division in the spirit of [10], which transforms paraconsistent logic programs into monotonic logic programs. Then we can similarly apply an immediate consequences operator. A minimum paraconsistent well-founded model is guaranteed to exist.

We further provide a simple translation of Extended Logic Programs under $WFSX_p$ [2] into Paraconsistent Logic Programs. Other settings are easily embeddable into our framework, such as Probabilistic Logic Programs [15], Possibilistic Logic Programming [6], Hybrid Probabilistic Logic Programs [5], Generalized Annotated Logic Programs [13], and Fuzzy Logic Programs [24]. On the one hand, the merit of these translations is in their allowing to deal with negation, paraconsistency, and non-monotonic reasoning within the uncertain formalisms above. On the other hand, we can now study the behaviour of $WFSX_p$, and of the most important paraconsistent semantics for Extended Logic Programs when uncertain reasoning is introduced into them.

As Antitonic Logic Programs, we can see Paraconsistent Logic Programs as a natural extension of the Fitting's works [8] and [9]. However, the central point in these works is the logic programming syntax instead of considering arbitrary monotonic and antitonic functions in the bodies. To be absolutely fair, the article [9] introduces the notions of *attenuation operators* which can be viewed as arbitrary monotonic functions over bilattices.

As future work, we will generalize our structure to consider rules with more complex heads, where we can have, for instance, disjunctions of atoms. We also will consider particular instances of our framework to improve the understanding of properties of the concrete instances, and to compare these instances to existing work. An interesting point of research is the study of the various types of negation in our framework, specially if we allow for weak negation operators as well. This offers the opportunity for studying how the principle of coherence functions in such cases. The definition of derivation procedures is also envisaged.

Acknowledgements. João Alcântara is a PhD student supported by CAPES - Brasília, Brazil. The authors also thank the TARDE and FLUX projects, sponsored by MCT - Portugal.

References

1. J. J. Alferes, L. M. Pereira, and T. C. Przymusinski. Belief revision in non-monotonic reasoning and logic programming. *Fundamenta Informaticae*, 28(1 & 2):1–22, 1996.
2. C. V. Damásio. *Paraconsistent Extended Logic Programming with Constraints.* PhD thesis, Universidade Nova de Lisboa, 1996.
3. C. V. Damásio and L. M. Pereira. Antitonic logic programs. In T. Eiter and M. Truszczynski, editors, *Procs. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*, LNCS/LNAI, pages 748–759. Springer, 2001.
4. C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In S. Benferhat and P. Besnard, editors, *6th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'01)*, LNCS/LNAI. Springer, 2001.

5. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *Journal of Logic Programming*, 43(3):197–250, 2000.
6. D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In *Proc. of ICLP'91*, pages 581–598. MIT Press, 1991.
7. M. Fitting. Logic programming on a topological bilattice. *Fundamentae Mathematicae*, XI:209–218, 1988.
8. M. Fitting. Bilattices and the semantics of logic programming. *The Journal of Logic Programming*, 11:91–116, 1991.
9. M. Fitting. The family of stable models. *Journal of Logic Programming*, 17:197–225, 1993.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
11. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.
12. M. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
13. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *The Journal of Logic Programming*, 12(1, 2, 3 & 4):335–367, 1992.
14. R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *7th International Conference on Logic Programming*. MIT Press, 1990.
15. L. Lakshmanan and F. Sadri. On a theory of probabilistic deductive databases. *Theory and Practice of Logic Programming*, 1(1):5–42, 2001.
16. L. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *Knowledge and Data Engineering*, 13(4):554–570, 2001.
17. Jan Maluszyński and Aida Vitória. Towards rough datalog: embedding rough sets in prolog. In A. Skowron Accepted in: S. K. Pal, L. Polkowski, editor, *Rough-Neuro Computing*, AI Series. Springer-Verlag, 2002.
18. R. Ng and V. S. Subrahmanian. Probabilistic logic programming. *INFCTRL: Information and Computation (formerly Information and Control)*, 101, 1992.
19. Z. Pawlak. *Rough Sets. Theoretical aspects of reasoning about data*. Kluwer Academic Publishers, Dordrecht, 1991.
20. L. M. Pereira and J. J. Alferes. Well-founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conference on Artificial Intelligence*, pages 102–106, Wien-Austria, 1992. John Wiley & Sons.
21. E. Shapiro. Logic programs with uncertainties: A tool for implementing expert systems. In *Proc. IJCAI'83*, pages 529–532. William Kauffmann, 1983.
22. M. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 3(1):37–53, 1986.
23. M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 4(23):733–742, 1976.
24. P. Vojtás and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In *Proc. of the Ws. on Extensions of Logic Programming (ELP'96)*, LNCS 1050, pages 289–301. Springer-Verlag, 1996.
25. G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *Mathematical Foundations of Database Systems*, pages 357–371. LNCS 495, Springer-Verlag, 1991.