Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Applied Logic 3 (2005) 67–95

JOURNAL OF APPLIED LOGIC

www.elsevier.com/locate/jal

An encompassing framework for Paraconsistent Logic Programs

João Alcântara *, Carlos Viegas Damásio, Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA), Departamento de Informática, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal

Available online 2 September 2004

Abstract

We propose a framework which extends Antitonic Logic Programs [Damásio and Pereira, in: Proc. 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning, Springer, 2001, p. 748] to an arbitrary complete bilattice of truth-values, where belief and doubt are explicitly represented. Inspired by Ginsberg and Fitting's bilattice approaches, this framework allows a precise definition of important operators found in logic programming, such as explicit and default negation. In particular, it leads to a natural semantical integration of explicit and default negation through the Coherence Principle [Pereira and Alferes, in: European Conference on Artificial Intelligence, 1992, p. 102], according to which explicit negation entails default negation. We then define Coherent Answer Sets, and the Paraconsistent Well-founded Model semantics, generalizing many paraconsistent semantics for logic programs. In particular, Paraconsistent Well-Founded Semantics with eXplicit negation ($WFSX_p$) [Alferes et al., J. Automated Reas. 14 (1) (1995) 93–147; Damásio, PhD thesis, 1996]. The framework is an extension of Antitonic Logic Programs for 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 simultaneously with default, paraconsistency, and uncertainty reasoning. Results are provided about how our semantical framework deals with inconsistent information and with its propagation by the rules of the program.

* Corresponding author.
E-mail addresses: jfla@di.fct.unl.pt (J. Alcântara), cd@di.fct.unl.pt (C.V. Damásio), lmp@di.fct.unl.pt (L.M. Pereira).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2004.07.012

Keywords: Paraconsistent reasoning; Logic programming; Multivalued logics; Bilattices; Knowledge representation

1. Introduction

The development of efficient tools to solve problems in knowledge representation urges a careful balance between their computational performance in carrying out some specific inference task and the attempt to find a precise formalization of the problem.

In virtue of their mathematical power, bilattices are strong enough to provide a uniform treatment of existing procedural systems, whilst keeping attractive computational properties. Introduced by Ginsberg [5], they have been used in the effort to implement knowledge representation tools. Furthermore, using bilattices in logic programming formalisms, Fitting attested (see [6–8]) that they are particularly adequate to represent knowledge in situations where we can find uncertainty, incompleteness, and inconsistency. The use of bilattices for supporting paraconsistent reasoning in logic programming has been supported by others, namely [9,10].

Fitting's results appeared at almost the same time as a diversity of frameworks for manipulating data and knowledge were proposed in the form of extensions to logic programming and deductive databases [8,11–14]. Usually, the authors characterize their programs with a model theoretical semantics, where a minimum model is guaranteed to exist, and a corresponding monotonic fixed point operator too (whether continuous or not).

The underlying uncertainty formalism in the proposed logic programming frameworks includes probability theory [15,16], fuzzy set theory [17,18], many-valued logic [7,14,19], and possibilistic logic [11]. Different ways of dealing with uncertainty may be required for any given application [20]. All such logic programming based frameworks are monotonic, and so none allows default negation.

Following an algebraic approach to both the language and the semantics of logic programs, Damásio and Pereira define in [21] 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 default negation) to arbitrary complete lattices of truth-values, via an appropriate notion of implication. It is shown that Monotonic Logic Programs are general enough to capture several distinct logic programming semantics such as the uncertainty formalisms above. In [1], the same authors generalize the framework to cater for rules with arbitrary antitonic bodies over general complete lattices, and show all standard logic programming theoretical results carry over to such Antitonic Logic Programs, defining them for Stable [22,23] and Well-founded Model [24] semantics alike.

Notwithstanding, a specific treatment for explicit negation in Antitonic Logic Programs is not provided. In the present work we extend the previous framework to an arbitrary complete bilattice of truth-values with appropriate negation and conflation operators. The resultant framework is hereby dubbed Paraconsistent Logic Programs. In its presentation we are motivated by Ginsberg's bilattices [5], and by Lakshmanan and Sadri's work on probabilistic deductive databases [15]. Ginsberg's bilattices support an elegant framework for logic programming involving belief and doubt [7]. In particular, they lead to a precise

definition of explicit negation operators. We further employ these results to properly characterize default negation, and ensure obedience to the coherence principle, to the effect that explicit negation entails default negation (i.e. whatever is explicitly false must necessarily be false by default as well). In [15], the authors argue about the convenience of explicitly representing both belief and doubt when dealing with incomplete knowledge, where diverse evidence may contradict one another.

Considering that various degrees of contradictory information can be found in Paraconsistent Logic Programs, besides satisfying the coherence principle, a semantics for these programs must be able to deal with both contradiction and uncertain information. Moreover, it is important not just to reason paraconsistently when facing an inconsistency, but also to keep track of which part of the knowledge base is itself inconsistent, which part merely depends on the inconsistent part and, of course, to detect too which part is inconsistency free. In order to define a semantics with these requirements, we generalize the paraconsistent well-founded semantics for extended logic programs $WFSX_p$ [3,4]. As we do not impose any specific characterization of explicit negation, we can introduce in our framework any negation operator supported by Ginsberg's bilattice. In the sequel, we present as well a semantics based on Coherent Answer Sets.

The paper is structured as follows: in Section 2 we present bilattices [5,7], and empower the framework to characterize the default negation operator. Sections 3 and 4 present the core of our work: there we introduce respectively syntax and semantics of Paraconsistent Logic Programs. In Section 5, we produce an illustrative example where the framework is utilized to encode a rather complex decision table. In the subsequent section, we demonstrate results concerning how to identify inconsistent propositional symbols or which depend on the inconsistency of other propositional symbols. Finally, we draw out conclusions, refer related work, and mention future developments. An extended version of this paper with the full proofs of all results is available at [25].

2. Bilattices

With the aim of characterizing uncertainty, several varieties of fixed point semantics for logic programming have been proposed. In common they require a suitable machinery ensuring the existence of fixed points of the particular 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 order of truth-values. It is claimed by Fitting [7] that the notion of bilattice, as presented by Ginsberg [5], can be applied to most kinds of fixed point semantics, providing an account of the intended partial order for each of the truth-functional connectives and their interactions.

By employing bilattices, we avail ourselves of a powerful tool to uniformly describe these truth-values, which can not only depict classical bivalued systems, but basically also any multivalued approach whose underlying truth-values have some intuitive character. As a matter of fact, in its most general presentation, bilattices can furthermore describe many non-intuitive truth-values [5].

Definition 1 (*Bilattice*). A bilattice is a structure $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ where B is a nonempty set, and $\langle B, \leqslant_k \rangle$ and $\langle B, \leqslant_t \rangle$ are both bounded lattices, i.e. with a bottom and a top element.

We shall use $\otimes_k$ and $\oplus_k$ respectively for denote the meet and join operations that correspond to $\leqslant_k$, and $\otimes_t$ and $\oplus_t$ for those that correspond to $\leqslant_t$.

The partial order $\leqslant_k$ is intended to represent the knowledge or information order, and $\leqslant_t$, the truth order. In other words, we can say the knowledge order reports on how much information we have about a particular statement p, while the truth order reports on how confident we are in that p is true or otherwise false. Interpreting $x \leqslant_t y$, we simply thereby mean y is truer than x; in turn, we interpret $x \leqslant_k y$ as meaning the evidence underlying x is subsumed by the evidence underlying y.

The bilattice $\mathcal{B}$ will be (infinitary) *k-distributive* if the lattice $\langle B, \leqslant_k\rangle$ is (infinitary) distributive; it will be (infinitary) *t-distributive* if $\langle B, \leqslant_t\rangle$ is (infinitary) distributive; we will say $\mathcal{B}$ is *cross distributive* if each of $\otimes_t$ and $\oplus_t$ distributes with respect to both $\otimes_k$ and $\oplus_k$. If $\mathcal{B}$ is k-distributive, t-distributive and cross distributive, we will simply say it is *distributive*. If both $\langle B, \leqslant_k\rangle$ and $\langle B, \leqslant_t\rangle$ are complete lattices, i.e. their respective least upper bound and greatest lower bound exist for arbitrary subsets of the lattice (not only for finite subsets), we say that the bilattice is *complete*.

We may sometimes refer to a bilattice as $\mathcal{B}(C, D)$ whenever we wish to emphasize its domain is the Cartesian product $C \times D$, where $\langle C, \leqslant_1\rangle$ and $\langle D, \leqslant_2\rangle$ give us the structure of complete lattices, and for any $\langle c_1, d_1\rangle$ and $\langle c_2, d_2\rangle$ elements of $C \times D$:

- $\langle c_1, d_1\rangle \leqslant_k \langle c_2, d_2\rangle$ iff $c_1 \leqslant_1 c_2$ and $d_1 \leqslant_2 d_2$,
- $\langle c_1, d_1\rangle \leqslant_t \langle c_2, d_2\rangle$ iff $c_1 \leqslant_1 c_2$ and $d_2 \leqslant_2 d_1$.

We shall use $\bot_1$ and $\top_1$ in order to denote, respectively, the bottom and top elements of $\langle C, \leqslant_1\rangle$; similarly $\bot_2$ and $\top_2$ will denote, respectively, the bottom and top elements of $\langle D, \leqslant_2\rangle$. When $\leqslant_1$ and $\leqslant_2$ are the same, we simply designate these elements by $\bot$ and $\top$. In a bilattice $\mathcal{B}(C, C)$ with $\leqslant_1 = \leqslant_2$, $\langle\bot, \bot\rangle$ and $\langle\top, \top\rangle$ are respectively the bottom and top elements of the bilattice with respect to $\leqslant_k$; likewise, $\langle\bot, \top\rangle$ and $\langle\top, \bot\rangle$ are respectively the bottom and top elements of the bilattice with respect to $\leqslant_t$.

We can imagine the pair $\langle c, d\rangle$ in $\mathcal{B}(C, D)$ represents two independent judgements concerning the truth of some statement: c represents our degree of evidence *for* the statement; d represents our degree of evidence *against* it. Since C and D can be the domain of different lattices, expressions of belief in and against need not be measured in the same way. In this sense, by $\langle c_1, d_1\rangle \leqslant_k \langle c_2, d_2\rangle$ we mean $\langle c_2, d_2\rangle$ embodies more "knowledge" than $\langle c_1, d_1\rangle$, which is reflected by an increased degree of both belief in and against. On other hand, by $\langle c_1, d_1\rangle \leqslant_t \langle c_2, d_2\rangle$ we mean $\langle c_2, d_2\rangle$ embodies more "truth" than $\langle c_1, d_1\rangle$, which is reflected in an increased degree of evidence for, and a decreased degree of evidence against it.

The simplest example of a non-trivial bilattice can be obtained by resorting to the lattice constituted by the set $\{0, 1\}$ with $0 \leqslant 1$ representing, respectively, the classical truth values *true* and *false*. The bilattice $\mathcal{B}(\{0, 1\}, \{0, 1\})$, depicted in Fig. 1, gives us an isomorphic copy of the well-known four-logic, due to Belnap [26], where $\langle 0, 0\rangle$ indicates we have no evidence either for or against, and $\langle 1, 1\rangle$ indicates we are in an inconsistent situation of having full evidence both for and against. Likewise, $\langle 0, 1\rangle$ stands for false and $\langle 1, 0\rangle$ for true.

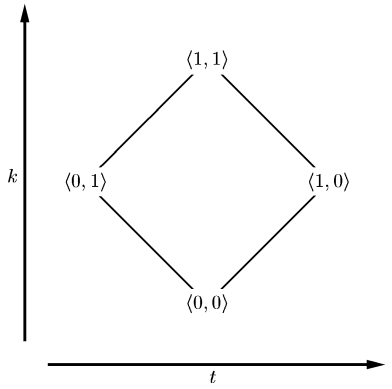

Fig. 1. The smallest non-trivial bilattice.

Analyzing the bilattice $\mathcal{B}(\{0,1\},\{0,1\})$ in terms of $\leqslant_k$, we can conclude that $\langle 0,0\rangle$ and $\langle 1,1\rangle$ are respectively the least and the greatest element, and $\langle 0,1\rangle$ and $\langle 1,0\rangle$ are incomparable. Considering the relation $\leqslant_t$, $\langle 0,1\rangle$ and $\langle 1,0\rangle$ are respectively the least and the greatest element, and $\langle 0,0\rangle$ and $\langle 1,1\rangle$ are incomparable.

Another well-known example is based on the real unit interval $[0,1]$. Taking C and D as $[0,1]$, we can think of a member of the resulting bilattice $\mathcal{B}([0,1],[0,1])$ as expressing a *degree* of evidence for and against. For more expressive useful bilattices, the reader is invited to take a look in [5–7,27].

Observe the only difference between our bilattice definition and Ginsberg's is that, like Fitting [7], we do not consider negation as granted. Thus we may have bilattices without a negation operator. In order to introduce it, Ginsberg [5] requires that negation mesh well with the partial orders and satisfy the double negation property:

Definition 2 (*Negation*). A bilattice $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k\rangle$ has a negation operation if there is a mapping $\neg : B \to B$ such that:

(1) $a \leqslant_k b \Rightarrow \neg a \leqslant_k \neg b$;
(2) $a \leqslant_t b \Rightarrow \neg b \leqslant_t \neg a$;
(3) $\neg\neg a = a$.

Due to the way negation is contemplated in a bilattice, we can argue this operator establishes a connection between the knowledge and truth orders in a reasonable way: one expects negation to invert the notion of truth, whilst negation would preserve the knowledge order, that is, one would know as much about $\neg p$ as one would know about p.

Suppose we have combined two identical lattices $\mathcal{C} = \langle C, \leqslant\rangle$ in order to create a bilattice $\mathcal{B}(C,C)$. A reasonable candidate then for the negation operator is $\neg\langle a,b\rangle = \langle b,a\rangle$. Intuitively, in a bilattice $\mathcal{B}(C,C)$, we are assuming evidence for and against are measured in the same way. It being so, in passing from $\langle a,b\rangle$ to $\neg\langle b,a\rangle$, we are just counting now "for" what was counted "against" before, and conversely.

Then, in the bilattice $\mathcal{B}(\{0,1\},\{0,1\})$ of Fig. 1, we can define the negation operator by reversing evidence for and against. This is equivalent to "flipping" the diagram left to right by interchanging $\langle 0,1\rangle$ and $\langle 1,0\rangle$ while keeping fixed $\langle 0,0\rangle$ and $\langle 1,1\rangle$.

Fitting [7] introduced one more basic operator in bilattices: conflation (see [7] for details). This operator is defined in the same manner as negation, but with the roles of $\leqslant_k$ and $\leqslant_t$ interchanged:

Definition 3 (*Conflation*). A bilattice $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k\rangle$ enjoys a conflation operation if there is a mapping $-: B \to B$ such that:

(1) $a \leqslant_k b \Rightarrow -b \leqslant_k -a$;
(2) $a \leqslant_t b \Rightarrow -a \leqslant_t -b$;
(3) $--a = a$.

If $\neg\neg a = a$ (respectively $--a = a$) is not verified, the negation (respectively conflation) operator is said to be weak. In [7], Fitting further states the conflation operator results in moving to "default" evidence. This signifies that given $L \in B$, where $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k\rangle$ is a bilattice with a conflation operator, in $-L$ we are to count now as "for" whatever did not count as "against" before, and "against" what did not count as "for".

Then suppose again we combine two identical lattices $\mathcal{C} = \langle C, \leqslant\rangle$ in order to create a bilattice $\mathcal{B}(C,C)$; if $\mathcal{C}$ has an order reversing involution (we denote the involute of x by $-x$), a reasonable candidate for the conflation operator is $-\langle a,b\rangle = \langle -b,-a\rangle$ [27]. The operator – is called a *de Morgan complement*. Recalling the bilattice $\mathcal{B}(\{0,1\},\{0,1\})$, we can introduce conflation by defining an operation which "flips" the bilattice in Fig. 1 from top to bottom, interchanging $\langle 0,0\rangle$ and $\langle 1,1\rangle$ while keeping fixed $\langle 0,1\rangle$ and $\langle 1,0\rangle$.

Under these circumstances, the notion of conflation is used by Fitting [7] to define important relations over the bilattice:

Definition 4. In a bilattice $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k\rangle$ with conflation, for $L \in B$

1. L is exact if $L = -L$;
2. L is consistent if $L \leqslant_k -L$.

Additionally, we will say L is inconsistent if $L \not\leqslant_k -L$.

Regarding bilattices of the kind $\mathcal{B}(C,D)$, we can think of the evidence for $-L$ as the complement of the evidence against L. Then we can say L is *exact* whenever evidence for and against complement each other in some reasonable sense. If L is exact or the evidence for (respectively evidence against) L has less knowledge than the complement of evidence against (respectively evidence for) L, we have that L is *consistent*. Otherwise, if there is a conflict of knowledge between evidence for and evidence against, i.e. the evidence for (respectively evidence against) L has more knowledge than the complement of evidence against (respectively evidence for) L, we have that L is *inconsistent*.

In the bilattice $\mathcal{B}([0,1],[0,1])$, for instance, we can define conflation as follows: $-\langle a,b\rangle = \langle 1-b, 1-a\rangle$. Considering this case, an element $\langle \alpha,\beta\rangle$ which belongs to

$[0,1] \times [0,1]$ is exact if $\alpha + \beta = 1$; it is consistent if $\alpha + \beta \leqslant 1$, and it is inconsistent if $\alpha + \beta > 1$.

Assuming the existence of conflation, associated to each negation operator we can define a default operator. In the same way, conflation represents a movement to default evidence, such a default operator, denoted by *not* L, would represent a move to default negation evidence. This means that given $L \in B$, where $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ is a bilattice with conflation and negation operators, in *not*L we are to count as "for" whatever did not count as "for" before, and "against" what did not count before as "against".

Assuming bilattice $\mathcal{B}$ with both $\neg$ and $-$, we say $\neg$ and $-$ *commute* if $\neg -A = -\neg A$ for all A. Taking such considerations into account we may generalize this notion of default negation as follows:

Definition 5 (*Default negation*). Let $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ be a bilattice. Consider that $\neg$ and $-$, respectively, a negation and a conflation operator, that commute in $\mathcal{B}$. We define *not* : $B \to B$ as the default negation operator where

$$not\, L =_{def} -\neg L.$$

In the bilattice $\mathcal{B}(\{0,1\},\{0,1\})$ (the crisp case), we can introduce default negation by defining an operation which "flips" the bilattice in Fig. 1 both top to bottom and left to right. This is equivalent to interchanging $\langle 0,0\rangle$ and $\langle 1,1\rangle$ as well as $\langle 0,1\rangle$ and $\langle 1,0\rangle$. As for the bilattice $\mathcal{B}([0,1],[0,1])$, we can define the default negation operator easily: $not\langle \alpha, \beta\rangle = \langle 1-\alpha, 1-\beta\rangle$. A different definition can be found in [10]. Default negation is *antitonic*[1] with respect to both orders, and thus preserves neither the knowledge order nor the truth order. In addition, this definition of *not* can be related to that of epistemic logic in [28], where *not* L is equated with $\mathbf{B} \sim L$, that is belief in $\sim L$ where $\sim$ is classical negation. There, $\mathbf{B}\sim L$ is given the semantics that $\sim L$ is true in all minimal models. To see the relationship, let L in *not* L above be represented by $\langle \alpha, \beta\rangle$, and let the bilattice be exact, that is $-\neg L = \neg L$. So $\langle 1-\alpha, 1-\beta\rangle = \langle \beta, \alpha\rangle$, i.e. α and β are the exact complements of one another, and consequently $\neg L$ can be interpreted as $\sim L$. It follows that in this case the conflation operator coincides indeed with the belief operator $\mathbf{B}$, since $not\, L = -\neg L = -\sim L = \mathbf{B} \sim L$. Thus one may conclude that conflation, as introduced in conjunction with *not*, further generalizes the notion of belief to bilatices.

The default negation operator plays a central role in our framework. In Section 3, following ideas expressed in [2], we will relate it to the explicit negation operator through the *coherence principle*. In its original concept, the coherence principle was introduced in Extended Logic Programs interpreted by $WFSX_p$.

The rationale of $WFSX_p$ is to non-trivially extract the maximum number of conclusions from contradictory information. This provides the user with the information necessary to decide what to do, since all possible scenarios are taken into account. The user is warned about some potential problems, and is up to him to take the right decision. In this work, we will generalize to some extent this property involving the detection of information support

[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.

on inconsistency, in order to make it applicable in complete bilattices of truth-values. For a recent characterization of $WFSX_p$ in terms of argumentation semantics, the reader is referred to [29].

As we have seen, bilattices provide a truth-value mechanism suitable for uncertain/inconsistent information. In Fitting's words [6], a logic programming fixed point semantics can be developed relative to any bilattice. That means that under the bilattice umbrella we can develop a framework general enough to capture a huge number of extant formalisms involving uncertain and/or inconsistent reasoning. In the next sections the presentation and motivation for this framework will be the central topic of our concern.

3. Paraconsistent Logic Programs

In the framework of Monotonic Logic Programs [21], rules are constituted by arbitrary isotonic body functions, and by propositional symbols in the head. In [1], 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 [22,23] and Well-founded [24] 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., [2,23,30,31]) have argued for 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 [10,23, 32,33] add explicit negation to normal programs. In [2], it is claimed that explicit negation should entail default negation, the *Coherence Principle* already mentioned before. Unfortunately, antitonic logic programs as they stand are not suited to characterize this principle, as discussed below.

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, that is, extended logic programs and Lakshmanan and Sadri's framework, we shall extend the syntax of antitonic logic programs to bodies constructed from combinations of functions isotonic or antitonic in each argument over a complete bilattice, and to more general head formulas. Then we employ the characterization of the default negation operator presented in Section 2 to impose coherence on it.

In the remainder of this work, we assume that every bilattice is complete, admits conflation and negation, and that these two operators commute. For the sake of completeness, we start by defining an algebraic characterization of the syntax of paraconsistent logic programs. The following presentation is rather standard and more detailed accounts can be found, for instance, in [34,35]. Since the language of our programs is single sorted, we resort to a simplified version of signature in specifying paraconsistent logic programs.

Definition 6 (*PLP signature*). A signature Σ is a set of pairs $\langle f, n\rangle$ such that no function symbol f occurs in two distinct pairs and its arity or rank n is a natural number ($n \geqslant 0$). The set Σ_n denotes the set of function symbols of arity n.

A paraconsistent logic program signature, PLP signature for short, is a signature containing at least the pairs $\langle \neg, 1\rangle$, $\langle -, 1\rangle$ and $\langle \otimes_k, 2\rangle$.

So, paraconsistent logic programs will be built using function symbols that represent at least the negation and conflation operators, and the meet under knowledge ordering. The user is free to add other function symbols and constants to the language, defining a specific PLP signature. The association of meaning to the symbols of a PLP signature is captured by the definition of PLP algebra:

Definition 7 (*PLP algebra*). Given a PLP signature Σ and a bilattice $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k\rangle$, a PLP algebra $\mathfrak{A}_\Sigma^{\mathfrak{B}}$ is a pair $\langle B, i\rangle$ where i is an interpretation function assigning functions to functions symbols as follows:

(1) Each symbol c in Σ_0 (i.e., c is a constant symbol) is interpreted as an element $i(c)$ in B, denoted by $\overset{\bullet}{c}$.
(2) Each function symbol f in Σ_n, $n \geqslant 1$, is interpreted as a function $i(f): B^n \to B$, denoted by $\overset{\bullet}{f}$. Additionally, $\overset{\bullet}{f}$ must be either isotonic or antitonic in each argument with respect to $\leqslant_k$.

Moreover, $i(\neg)$ and $i(-)$ must be respectively mapped to the negation and conflation operators of $\mathfrak{B}$, and $i(\otimes_k)$ to the meet operator under knowledge ordering of $\mathfrak{B}$.

Note that PLP algebras are related to semantical aspects of the operators. The next step consists in defining the alphabet of symbols of our language, given a PLP signature.

Definition 8 (*PLP alphabet*). Let Σ be a PLP signature and $\Phi = \bigcup_i \Sigma_i$ $(i \in \mathbb{N})$, the set of function symbols in Σ. Given a set Π of propositional symbols such that $\Pi \cap \Phi = \{\}$, the PLP alphabet $\mathcal{A}_\Sigma^\Pi$ over Π is the disjoint union $\Pi \cup \Phi$.

The formulae which appear in the body and heads of rules in paraconsistent logic programs are defined inductively as follows:

Definition 9 (*PLP formulae*). Let Σ be a signature and Π a set of propositional symbols. The set of PLP formulae $FORM_\Sigma^\Pi$ is the least set of strings over the corresponding PLP alphabet $\mathcal{A}_\Sigma^\Pi$ such that

- Every propositional symbol in Π is a formula in $FORM_\Sigma^\Pi$;
- Every constant symbol $c \in \Sigma_0$ is a formula in $FORM_\Sigma^\Pi$;
- If $F_1, \ldots, F_n$ $(n \geqslant 1)$ are formulae in $FORM_\Sigma^\Pi$ and f is a n-ary function symbol $(f \in \Sigma_n)$, then $f\ F_1\ \ldots\ F_n$ is a formula in $FORM_\Sigma^\Pi$.

In order to avoid unnecessary notational overhead, in this definition we specify formulae in prefix notation. It can be shown that the set of formulae $FORM_\Sigma^\Pi$ is freely generated from the propositional and constant symbols, and the operations corresponding to function

symbols, as described above. The notion of PLP formulae mimics the definition of first-order logic terms in prefix notation, with the predefined unary functions symbols $\neg$ and $-$, and the meet operator symbol $\otimes_k$. A complete formal analysis and justification of the above definitions can be found for instance in [34].

Before introducing the syntax of paraconsistent logic programs we require the notion of reverse functions. A major motivation for this definition is the use of negation operators in the head of rules. As we shall see, the bodies of our rules can be arbitrary formulae but we have to limit the allowed functions in the heads in order to guarantee the existence of a single model for every program under our Paraconsistent Well-Founded Semantics. As we shall see, the notion of reverse function allows moving functions in the head to the body.

Definition 10 (*Reverse functions*). Given an isotonic unary function $f : B \to B$ and a partial order $\leqslant$, we say f has a reverse function with respect to $\leqslant$ iff there is an isotonic function $\overline{f} : B \to B$ with respect to $\leqslant$, defined as follows: for all $b \in B$, $\overline{f}(b) = a$, where a is the minimum element according to $\leqslant$ such that $b \leqslant f(a)$, formally:

$$\overline{f}(b) = \min\{a \mid b \leqslant f(a)\}.$$

It is obvious that if $\overline{f}$ exists it will be unique. The pair $(f, \overline{f})$ is a Galois connection in the sense of [36], then it follows that $\overline{f}(b) \leqslant a$ iff $b \leqslant f(a)$.

Proposition 11. *Let $\neg$ be the negation operator of a bilattice. The reverse function of $\neg$ with respect to the knowledge ordering is $\neg$ itself.*

As examples of operators with reverse functions we will only consider negation in the bilattice. However, with a view towards generality, we foresee from the start the possibility of introducing other operators with this property, for instance special instances of weak negations. The full syntax of Paraconsistent Logic Programs can now be proffered:

Definition 12 (*Paraconsistent Logic Programs*). Let Σ be a PLP signature, $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ a bilattice and $\mathfrak{A}_\Sigma^{\mathfrak{B}}$ a PLP algebra. A paraconsistent logic program P over the set of propositional symbols Π is a set of rules of the form $\Phi[A] \leftarrow \Psi$ where $\Phi[A]$ and Ψ are formulas in $FORM_\Sigma^\Pi$, such that:

(1) The head $\Phi[A]$ has the form $o_1\ o_2\ \ldots\ o_n\ A$ $(n \geqslant 0)$, i.e. a possible empty sequence of unary function symbols applied to propositional symbol A.
(2) Each o_i occurring in $\Phi[A]$ is an isotonic unary operator symbol in Σ_1 for which there is a $q_i \in \Sigma_1$ such that $\overset{\bullet}{q_i}$ is a reverse function of $\overset{\bullet}{o_i}$ with respect to $\leqslant_k$ in $\mathfrak{B}$.
(3) The body Ψ is an arbitrary formula of $FORM_\Sigma^\Pi$.

If $\Phi[A] = o_1\ o_2\ \ldots\ o_n\ A$ occurs in the head of a rule and F is an arbitrary formula of $FORM_\Sigma^\Pi$, we denote by $\overline{\Phi}[F]$ the formula $q_n\ \ldots\ q_2\ q_1\ F$, obtained by concatenating $q_n\ \ldots\ q_2\ q_1$ with F. Mark that if $\Phi[A] = A$ then $\overline{\Phi}[F] = F$.

Throughout this work, we reserve the symbol $\mathfrak{R}$ to denote the PLP algebra below:

Example 13. Let Σ be the following PLP signature:

$$\Sigma = \big\{\langle\neg,1\rangle,\ \langle-,1\rangle\langle\otimes_k,2\rangle\big\} \cup \big\{\langle\langle a,b\rangle,0\rangle \mid a,b\in[0,1]\big\},$$

where the required function symbols of a PLP signature are extended with constants representing all the elements in the set $[0,1]^2$. Notice that in the above expression $\langle a,b\rangle$ is interpreted as a single constant symbol with no internal structure. Instead, we could have used something like n^{α}_{β}, but that would compromise readability of the examples.

Consider the bilattice $\mathcal{B}([0,1],[0,1])$ constructed from the Cartesian product of the unit interval $[0,1]$ with itself, and the usual total ordering among real numbers.

The PLP algebra $\mathfrak{R}$ is the pair $\langle[0,1]^2, i\rangle$ constructed from $\mathcal{B}([0,1],[0,1])$ such that

(1) $i(\langle a,b\rangle) = \langle a,b\rangle$, i.e. all the constant symbols are mapped to the corresponding elements in $[0,1]^2$.
(2) $i(\neg):[0,1]^2\to[0,1]^2$ such that $i(\neg)(\langle\alpha,\beta\rangle) = \langle\beta,\alpha\rangle$.
(3) $i(-):[0,1]^2\to[0,1]^2$ such that $i(-)(\langle\alpha,\beta\rangle) = \langle 1-\beta, 1-\alpha\rangle$.
(4) $i(\otimes_k):[0,1]^2\times[0,1]^2\to[0,1]^2$ such that $i(\otimes_k)(\langle\alpha,\beta\rangle,\langle\gamma,\delta\rangle) = \langle\min(\alpha,\gamma), \min(\beta,\delta)\rangle$, where the function $\min:[0,1]\times[0,1]\to[0,1]$ returns the minimum of its arguments.

Considering the knowledge ordering in $\mathcal{B}([0,1],[0,1])$, the function $i(\neg)$ is isotonic, and $i(\otimes_k)$ is isotonic in both arguments whilst $i(-)$ is antitonic. Recall that we can use the alternative notations $\overset{\bullet}{\neg}$, $\overset{\bullet}{-}$, and $\overset{\bullet}{\otimes}_k$ for denoting $i(\neg)$, $i(-)$, and $i(\otimes_k)$, respectively.

Assuming the set of propositional symbols Π contains $\{a,b,c,d,e\}$, a syntactically correct paraconsistent logic program is:

$$c \leftarrow \otimes_k\ \otimes_k\ \otimes_k\ a\ \neg b\ a{-}\neg b \qquad \neg a \leftarrow \langle 0.8, 0.6\rangle \qquad e \leftarrow -\neg d$$
$$b \leftarrow \otimes_k\ \neg\neg a\ \langle 0.9, 1.0\rangle \qquad d \leftarrow -\langle 1.0, 0.0\rangle \qquad e \leftarrow \neg d$$

In order to improve readability, we shall serve our programs with rule bodies represented in infix notation, where the operator $\otimes_k$ associates to the left, and the negation and conflation operators have higher priority than $\otimes_k$. With these usual conventions, all examples presented in this paper are unambiguously translated to prefix notation as required by our formal definitions. Thus, the above program may be exhibited as follows:

$$c \leftarrow a\ \otimes_k\ \neg b\ \otimes_k\ a\ \otimes_k\ -\neg b \qquad \neg a \leftarrow \langle 0.8, 0.6\rangle \qquad e \leftarrow -\neg d$$
$$b \leftarrow \neg\neg a\ \otimes_k\ \langle 0.9, 1.0\rangle \qquad d \leftarrow -\langle 1.0, 0.0\rangle \qquad e \leftarrow \neg d$$

The following definition is tailored with the aim of ascertaining if an occurrence of a propositional symbol is isotonic or antitonic:

Definition 14 (*Isotonic and antitonic occurrences*). Consider a set of propositional symbols Π and let $\Pi^{\pm}$ be the set of annotated propositional symbols $\Pi^{\pm} = \{A^{+} \mid A\in\Pi\}\cup\{A^{-} \mid A\in\Pi\}$. The functions $^{\mp}:FORM^{\Pi}_{\Sigma}\to FORM^{\Pi^{\pm}}_{\Sigma}$ and $^{\mp}:FORM^{\Pi}_{\Sigma}\to FORM^{\Pi^{\pm}}_{\Sigma}$ are defined inductively by:

- For a propositional symbol A, $(A)^{\pm} = A^{+}$ and $(A)^{\mp} = A^{-}$.
- For a constant symbol c, $(c)^{\pm} = c$ and $(c)^{\mp} = c$.
- For every function symbol f of arity $(n \geqslant 1)$ and for all formulae $F_1, \ldots, F_n$ in $FORM_{\Sigma}^{\Pi}$, then

$$(f\ F_1\ \ldots\ F_n)^{\pm} = f\ G_1\ \ldots\ G_n \quad \text{and} \quad (f\ F_1\ \ldots\ F_n)^{\mp} = f\ H_1\ \ldots\ H_n$$

such that for every $1 \leqslant i \leqslant n$, formula G_i is $F_i^{\pm}$ and H_i is $F_i^{\mp}$ if $\overset{\bullet}{f}$ is isotonic in the ith argument, and G_i is $F_i^{\mp}$ and H_i is $F_i^{\pm}$ if $\overset{\bullet}{f}$ is antitonic in the ith argument.

A propositional symbol A annotated with A^{+} in $\Psi^{\pm}$ is said to be an isotonic occurrence of A in Ψ. Similarly, if A is annotated with A^{-} in $\Psi^{\pm}$ then the occurrence is antitonic in Ψ.

It should be clear that every occurrence of a propositional symbol in Ψ appears in the translated formula $\Psi^{\pm}$ annotated either with $^{+}$ or $^{-}$, and thus will be either isotonic or antitonic. The designations isotonic and antitonic occurrence will be fully understood after defining how a formula is "evaluated" with respect to a partial interpretation. For instance, $(\otimes_k\ \otimes_k\ \otimes_k\ a\ \neg b\ a\ -\neg b)^{\pm}$ is the formula $\otimes_k\ \otimes_k\ \otimes_k\ a^{+}\ \neg b^{+}\ a^{+}\ -\neg b^{-}$, as expected.

The expression $\Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n]$ $(m, n \geqslant 0)$ denotes the formula Ψ, where $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ are, respectively, the isotonic and antitonic occurrences, of propositional symbols in Ψ in the order they appear in Ψ (left to right). We usually denote a rule $\Phi[A] \leftarrow \Psi$ of P by $\Phi[A] \leftarrow \Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n]$ in order to make clear the isotonic and antitonic occurrences of propositional symbols in the body of the rule.

A paraconsistent logic program P is a monotonic logic program if all the rules are of the form $\Phi[A] \leftarrow \Psi[A_1, \ldots, A_m \mid]$. We refer to P as an antitonic logic program if each rule is either of the kind $\Phi[A] \leftarrow \Psi[A_1, \ldots, A_m \mid]$ or else $\Phi[A] \leftarrow \Psi[\ \mid B_1, \ldots, B_n]$.

4. Semantics of Paraconsistent Logic Programs

Regarding the semantics, we follow a paraconsistent and paracomplete approach inspired by $WFSX_p$ [3,4], one of the well-founded based semantics proposed for extended logic programs [37]. Furthermore, we introduce a Stable Model semantics too: the *Coherent Answer Sets*. The well-founded based semantics will be defined via an alternating fixed point definition in the style of [38], and also relates to the works in [9,27].

In order to simplify the presentation, we assume in the rest of this section the existence of a given PLP signature Σ, a PLP algebra $\mathfrak{A}_{\Sigma}^{\mathfrak{B}}$ for a complete bilattice $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ with conflation and negation commuting in $\mathfrak{B}$, and a set of propositional symbols Π.

The first difficulty in defining the semantics for paraconsistent logic programs concerns the evaluation of body formulae with antitonic occurrences of propositional symbols, in particular for defining default negation. A simple fixed point definition of the semantics is hampered because the bodies of rules have no more an isotonic (or monotonic) behavior with respect to an uniform assignment of elements in the bilattice to propositional symbols (an interpretation). A technique used in logic programming literature resorts to the notion of partial interpretation, requiring first the usual notion of interpretation.

Definition 15 (*Interpretation*). Let $\mathfrak{B} = \langle B, \leqslant_t, \leqslant_k \rangle$ be a complete bilattice and Π a set of propositional symbols. An interpretation of Π is a mapping $I : \Pi \to B$. The set of all interpretations of the propositional symbols with respect to $\mathfrak{B}$ is denoted by $\mathcal{I}_{\mathfrak{B}}$.

The pointwise ordering extends the knowledge ordering $\leqslant_k$ on the truth-values in $\mathfrak{B}$ to the set of interpretations as follows.

Definition 16 (*Lattice of interpretations*). Consider $\mathcal{I}_{\mathfrak{B}}$ the set of all interpretations of a set of propositional symbols Π with respect to a complete bilattice $\mathfrak{B}$, and two interpretations $I_1, I_2 \in \mathcal{I}_{\mathfrak{B}}$. Then, $\langle \mathcal{I}_{\mathfrak{B}}, \sqsubseteq \rangle$ is the complete lattice where $I_1 \sqsubseteq I_2$ iff $\forall_{p \in \Pi}\ I_1(p) \leqslant_k I_2(p)$. The least interpretation $\triangle$ maps every propositional symbol to the least element under $\leqslant_k$ of $\mathfrak{B}$, and the greatest interpretation $\triangledown$ maps every propositional symbol to the top element under $\leqslant_k$ of the same complete bilattice of truth-values $\mathfrak{B}$.

A partial interpretation is a pair of interpretations mapping propositional symbols to elements in the underlying bilattice.

Definition 17 (*Partial interpretations*). A partial interpretation I^p of a set of propositional symbols Π is a pair $I^p = \langle I^t, I^{tu} \rangle$ of interpretations of Π with respect to a complete bilattice $\mathfrak{B}$. The set of all partial interpretations is $\mathcal{I}^p_{\mathfrak{B}}$.

When speaking about (partial) interpretations we usually omit the set of propositional symbols Π, which is implicitly provided. Sometimes we refer to I^t as the T-component and to the I^{tu} as the TU-component of a partial interpretation. The T-component represents what certainly holds in the partial interpretation, while the TU-component contains what may hold (i.e. the "complement" of what certainly does not hold). The annotations T and TU are a remnant of the $WFSX_p$ semantics basis, since in a two-valued setting the T-component and TU-component capture, respectively, what is "true" and what is "true or undefined" ("non-false"). It is also important to mark that ($I^t \sqsubseteq I^{tu}$) is not imposed and, consequently, paraconsistency is allowed, i.e. something may certainly hold and not hold (by being simultaneously "true" and not "non-false"). Two orders among partial interpretations are useful:

Definition 18 (*Standard and fitting orderings*). Let I^p_1 and I^p_2 be two partial interpretations. The standard and Fitting's orderings among partial interpretations are defined by:

$$\textit{Standard ordering}: \quad I^p_1 \sqsubseteq_s I^p_2 \quad \text{iff} \quad I^t_1 \sqsubseteq I^t_2 \text{ and } I^{tu}_1 \sqsubseteq I^{tu}_2.$$
$$\textit{Fitting's ordering}: \quad I^p_1 \sqsubseteq_f I^p_2 \quad \text{iff} \quad I^t_1 \sqsubseteq I^t_2 \text{ and } I^{tu}_2 \sqsubseteq I^{tu}_1.$$

The set of partial interpretations ordered by $\sqsubseteq_s$ or by $\sqsubseteq_f$ is a complete lattice. Clearly, the bottom and top elements of these lattice are $\bot^p_s = \langle \triangle, \triangle \rangle$ (viz. all is false), $\top^p_s = \langle \triangledown, \triangledown \rangle$ (viz. all is true), $\bot^p_f = \langle \triangle, \triangledown \rangle$ (viz. all is undefined) and $\top^p_f = \langle \triangledown, \triangle \rangle$ (viz. all is true and false).

The standard ordering prefers partial interpretations maximizing what certainly holds and minimizing what certainly does not hold. Fitting's ordering prefers interpretations with

more information. The insight is that isotonic occurrences of propositional symbols (annotated with $^+$) should be evaluated in the T-component of the partial interpretation, while the antitonic occurrences (annotated with $^-$) should be evaluated in the TU-component, resulting in an element of the bilattice $\mathfrak{B}$. This separation between isotonic and antitonic occurrences of propositional symbols is fundamental to allow for the specification of converging operators in order to obtain fixed point semantics for every paraconsistent logic program [9,22,27,38]. A precise inductive definition of the evaluation of formula in a partial interpretation can now be provided:

Definition 19 (*Valuation*). For a given partial interpretation $I^p = \langle I^t, I^{tu}\rangle$ define inductively the function $val_{I^p} : FORM_\Sigma^{\Pi^\pm} \to B$ as follows:

- For every propositional symbol A, $val_{I^p}(A^+) = I^t(A)$.
- For every propositional symbol A, $val_{I^p}(A^-) = I^{tu}(A)$.
- For every constant symbol c, $val_{I^p}(c) = \overset{\bullet}{c}$.
- For every function symbol f of arity $(n \geqslant 1)$ and for all formulae $F_1, \ldots, F_n$ in $FORM_\Sigma^{\Pi^\pm}$, then $val_{I^p}(f\ F_1\ \ldots\ F_n) = \overset{\bullet}{f}(val_{I^p}(F_1), \ldots, val_{I^p}(F_1))$.

The valuation function $\widehat{I^p} : FORM_\Sigma^{\Pi} \to B$ is defined by $\widehat{I^p}(\Psi) = val_{I^p}(\Psi^\pm)$. If the formula Ψ is of the form $\Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n]$ by the expression $\widehat{\Psi}[I(A_1), \ldots, I(A_m) \mid J(B_1), \ldots, J(B_n)]$ we mean $\widehat{\langle I, J\rangle}(\Psi)$, where $\langle I, J\rangle$ is the partial interpretation constructed from the interpretations I and J.

The main property of the valuation function is captured in the following proposition, explaining why we designated isotonic and antitonic occurrences of propositional symbols those annotated with $^+$ and $^-$, respectively:

Proposition 20. *Consider the partial interpretations I^p and J^p with $I^p \sqsubseteq_f J^p$, then for every formula Ψ in $FORM_\Sigma^{\Pi}$:*

$$val_{I^p}(\Psi^\pm) \leqslant_k val_{J^p}(\Psi^\pm) \quad \text{and} \quad val_{I^p}(\Psi^\mp) \geqslant_k val_{J^p}(\Psi^\mp).$$

Therefore, $\widehat{I^p}(\Psi) \leqslant_k \widehat{J^p}(\Psi)$.

An increase in the values assigned to propositional symbols in the T-component and decrease of the values in the TU-component of a partial interpretation maintains or increases the value of a formula with respect to that partial interpretation. The notion of model of a program is now straightforward:

Definition 21 (*Model*). A partial interpretation I^p satisfies a rule $\Phi[A] \leftarrow \Psi$ of a paraconsistent logic program P iff $\widehat{I^p}(\Psi) \leqslant_k \widehat{I^p}(\Phi[A])$. A partial interpretation I^p is a model of P iff I^p satisfies all rules of P.

Notice we resort to the knowledge ordering to specify the notion of model. The intent of paraconsistent logic programs is to specify what should hold and not hold via program

rules, i.e. how information is extracted from program rules. This is the desired behavior in logic programming over bilattices, as argued in [10,27], since $\leqslant_k$ symmetrizes evidence for and against. The least upper bound of the knowledge ordering can be interpreted as an "accept all" operator that combines knowledge, in order to record cases in which there is an evidence at the same time for and against a given proposition, and pinpoint contradictory knowledge [10].

In particular, assume we state a proposition, say a, to be both true and false, via the rules $a \leftarrow \mathbf{t}$ and $\neg a \leftarrow \mathbf{t}$, where the constant $\mathbf{t}$ is mapped to the greatest element in the truth-ordering. Using the knowledge ordering, a is mapped to the greatest element in the knowledge ordering, meaning that there is some sort of inconsistency in the program. Using the truth-ordering instead, the rule $\neg a \leftarrow \mathbf{t}$ would be trivially satisfied, and a would be mapped to the greatest element in the truth-ordering, therefore losing the information that there is some sort of inconsistency in the program.

Proposition 22. *A partial interpretation satisfies a rule $\Phi[A] \leftarrow \Psi$ iff satisfies the rule $A \leftarrow \overline{\Phi}[\Psi]$.*

From the previous proposition we conclude that the function symbols appearing in the head can be "moved" to the body of the rule. Thus, function symbols in rule heads are simply "syntactic sugar" allowing the user to more naturally express his knowledge in a paraconsistent logic program.

The next major definition generalizes the Gelfond–Lifschitz transformation and the Γ operator [22]. The rationale is to extract from the program all its consequences, assuming that an interpretation is used to evaluate the antitonic occurrences of propositional symbols. Technically, this is achieved by coupling together the ideas of Gelfond–Lifschitz division operators [22] and of the immediate consequences operator of van Emden and Kowalski [39], along the lines of [27].

Definition 23 (*Gamma operator*). Let P be a paraconsistent logic program and J a given interpretation. The generalized immediate consequences operator $T^{\mathfrak{B}}_{\frac{P}{J}} : \mathcal{I}_{\mathfrak{B}} \to \mathcal{I}_{\mathfrak{B}}$ maps interpretations to interpretations:

$$T^{\mathfrak{B}}_{\frac{P}{J}}(I)(A) = lub_k\Big\{\widehat{\overline{\Phi}[\Psi]}[I(A_1), \ldots, I(A_m) \mid J(B_1), \ldots, J(B_n)] \\ \text{such that } \Phi[A] \leftarrow \Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n] \in P\Big\}.$$

The $\Gamma^{\mathfrak{B}}_P$ operator is defined as the least fixed point of $T^{\mathfrak{B}}_{\frac{P}{J}}$ with respect to the $\sqsubseteq$ ordering between interpretations, formally:

$$\Gamma^{\mathfrak{B}}_P(J) = \mathrm{lfp} T^{\mathfrak{B}}_{\frac{P}{J}} = T^{\mathfrak{B}}_{\frac{P}{J}} \uparrow^{\lambda}, \quad \text{for some ordinal } \lambda.$$

The definition of the generalized immediate consequences operator $T^{\mathfrak{B}}_{\frac{P}{J}}$ makes use of Proposition 22 guaranteeing the semantic equivalence of the rules $A \leftarrow \overline{\Phi}[\Psi]$ and $\Phi[A] \leftarrow \Psi$. The $T^{\mathfrak{B}}_{\frac{P}{J}}$ operator determines the value assigned to each propositional symbol

in order to satisfy all its rules, via the least upper bound under knowledge ordering (lub_k), for an interpretation of isotonic occurrences of propositional symbols and fixed interpretations of antitonic ones. The operator must be "iterated" from $\triangle$ in order to guarantee that all the rules for all the propositional symbols are satisfied, for the fixed interpretation of the antitonic occurrences of propositional symbols. This is attained by the $\Gamma_P^{\mathfrak{B}}$ operator.

Notwithstanding, in conformity with [2], negation and default negation are not unrelated: the Coherence Principle, when adopted, 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, i.e. its negation is false in all minimal models [28]. Therefore, one has $A \leqslant_k not\, \neg A$ (Coherence Principle). Since $not\, \neg A =_{def} -A$ (Definition 5), it suffices to guarantee $A \leqslant_k -A$ then. Technically, we can define the coherence principle in the following manner:

Definition 24 (*Coherence Principle*). Let P be a paraconsistent logic program. Then a model $M^p = \langle M^t, M^{tu}\rangle$ of P complies with the Coherence Principle iff for every propositional symbol A appearing in the language of P, $\widehat{M^p}(A) \leqslant_k \widehat{M^p}(-A)$, i.e. $M^t(A) \leqslant_k \overset{\bullet}{-} M^{tu}(A)$.

Given these arguments, and in order to enforce coherence, we will resort to the semi-normal gamma operator, inspired by the approach taken in the definition of the paraconsistent well-founded semantics with explicit negation $WFSX_p$ [3,4], itself inspired by the semi-normal gamma operator of $WFSX$ [40], which it generalizes to the paraconsistent case. Semi-normality itself is a designation borrowed from semi-normal defaults, the reason being that the semi-normal logic program rules add to the rule's body the default literal $not\, \neg H$, where H is a propositional symbol or their explicit negation in the head. In our context, $not\, \neg H$ is, by definition, $-H$.

Definition 25 (*Semi-normal Gamma operator*). Let P be a paraconsistent logic program and J an interpretation. The semi-normal immediate consequences operator $T^{\mathfrak{B}}_{\frac{P_s}{J}} : \mathcal{I}_{\mathfrak{B}} \to \mathcal{I}_{\mathfrak{B}}$ maps interpretations to interpretations:

$$T^{\mathfrak{B}}_{\frac{P_s}{J}}(I)(A) = lub_k\Big\{\widehat{\overline{\Phi}[\Psi]}[I(A_1), \ldots, I(A_m) \mid J(B_1), \ldots, J(B_n)] \overset{\bullet}{\otimes}_k \overset{\bullet}{-} J(A) \quad \text{such that } \Phi[A] \leftarrow \Psi[A_1, \ldots, A_m \mid B_1, \ldots, B_n] \in P\Big\}.$$

We also define $\Gamma^{\mathfrak{B}}_{P_s}(J) = \text{lfp}\, T^{\mathfrak{B}}_{\frac{P_s}{J}} = T^{\mathfrak{B}}_{\frac{P_s}{J}}\uparrow^{\lambda}$, for some ordinal λ.

Mark that coherence is enforced in every propositional symbol A by not letting the value of the bodies for A surpass $\overset{\bullet}{-} J(A)$. Before proceeding, it is necessary to ensure that both $\Gamma^{\mathfrak{B}}_P(J)$ and $\Gamma^{\mathfrak{B}}_{P_s}(J)$ are well defined. This is immediate from Knaster–Tarski theorem [41], since both versions of the immediate consequence operators are monotonic:

Theorem 26 (Monotonicity of $T^{\mathfrak{B}}_{\frac{P}{J}}$ *and* $T^{\mathfrak{B}}_{\frac{P_s}{J}}$). *Let P be a paraconsistent logic program and J an interpretation. The operators $T^{\mathfrak{B}}_{\frac{P}{J}}$ and $T^{\mathfrak{B}}_{\frac{P_s}{J}}$ are monotonic with respect to the pointwise extension $\sqsubseteq$ of knowledge ordering to interpretations.*

The dual alternating $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_P$ operator suffices to define an extension of well-founded semantics [24,38] to bilattice based logic programs, as done in [9,27]. In particular, this technique is followed in [1] to define semantics for Antitonic Logic Programs. Since $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_P$ does not take into account semi-normality, it cannot capture the Coherence Principle. The solution is described in [3,4], in which an alternating fixed point 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 ensure the next theorem:

Theorem 27 (Anti-monotonicity). *Consider a paraconsistent logic program P. Let J_1 and J_2 be two interpretations such that $J_1 \sqsubseteq J_2$. Then $\Gamma^{\mathfrak{B}}_P(J_2) \sqsubseteq \Gamma^{\mathfrak{B}}_P(J_1)$ and $\Gamma^{\mathfrak{B}}_{P_s}(J_2) \sqsubseteq \Gamma^{\mathfrak{B}}_{P_s}(J_1)$.*

Corollary 28. *Consider a paraconsistent logic program P. Let J_1 and J_2 be two interpretations such that $J_1 \sqsubseteq J_2$. Then $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}(J_1) \sqsubseteq \Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}(J_2)$.*

From the monotonicity of the $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}$ operators we know, again by the Knaster–Tarski theorem [41], for any paraconsistent logic program P, that $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}$ has a least fixed point. This fixed point is affirmed to define the paraconsistent well-founded semantics:

Definition 29 (*Paraconsistent well-founded semantics*). Let P be a paraconsistent logic program, and $M^p = \langle M^t, M^{tu} \rangle$ be a partial interpretation. We say M^p is a partial paraconsistent stable model of P iff $M^t = \Gamma^{\mathfrak{B}}_P(\Gamma^{\mathfrak{B}}_{P_s}(M^t))$ and $M^{tu} = \Gamma^{\mathfrak{B}}_{P_s}(M^t)$.

The least partial paraconsistent stable model under the Fitting ordering is the paraconsistent well-founded model $WFM^p(P)$, and can be obtained by iterating $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}$ from $\triangle$. Then, given that interpretation I_w is the least fixed point of $\Gamma^{\mathfrak{B}}_P \Gamma^{\mathfrak{B}}_{P_s}$ under $\sqsubseteq$ ordering, we shall have $WFM^p(P) = \langle I_w, \Gamma^{\mathfrak{B}}_{P_s}(I_w) \rangle$.

Given that $M^p = \langle M^t, M^{tu} \rangle$ is a partial paraconsistent stable model, we say a propositional symbol A is:

- *fully defined* with respect to M^p iff $M^t(A) = M^{tu}(A)$;
- *undefined* with respect to M^p iff $M^t(A) <_k M^{tu}(A)$;
- *overdefined* with respect to M^p iff $M^t(A) \not<_k M^{tu}(A)$.

We also say a propositional symbol A is strictly overdefined with respect to M^p iff $M^{tu}(A) <_k M^t(A)$.

Alternatively, by letting $\Omega^{\mathfrak{B}}_P(J) = \langle \Gamma^{\mathfrak{B}}_P(J^{tu}), \Gamma^{\mathfrak{B}}_{P_s}(J^t) \rangle$, an operator mapping partial interpretations to partial interpretations, the partial paraconsistent stable models of the program P can be seen as the fixed points of $\Omega^{\mathfrak{B}}_P$. Since $\Omega^{\mathfrak{B}}_P$ is monotonic with respect to $\sqsubseteq_f$,

the least fixed point under the Fitting ordering (the paraconsistent well-founded model) is once more guaranteed to exist, and can be obtained by iterating $\Omega_P^{\mathfrak{B}}$ from $\bot_f^p = \langle \triangle, \triangledown \rangle$. The $\Omega_P^{\mathfrak{B}}$ operator is the counterpart of the Przymusinski operator to characterize well-founded semantics [42].

An important result is that every partial paraconsistent stable model is indeed a model of the program:

Theorem 30. *Let P be a paraconsistent logic program. Every partial paraconsistent stable model of P is a model of P.*

When both components of the partial interpretation coincide, then an extension of answer set semantics [23] is obtained:

Definition 31 (*Coherent answer sets*). Let P be a paraconsistent logic program. A coherent answer set is a fully defined partial paraconsistent stable model, i.e. of the form $\langle M, M \rangle$, where M is an interpretation.

As usual, coherent answer sets are not guaranteed to exist. However, like paraconsistent well-founded models, coherent answer sets do comply with the Coherence Principle:

Proposition 32. *Let P be a paraconsistent logic program. Then all partial paraconsistent stable models $M^p = \langle M^t, M^{tu} \rangle$ of P obey the Coherence Principle, i.e. $\widehat{M^p}(A) \leqslant_k \widehat{M^p}(-A)$ for every propositional symbol A appearing in the language of P.*

The Coherence Principle is still valid for more general classes of formulae, as described in the following corollary:

Corollary 33. *Let P be a paraconsistent logic program, and $M^p = \langle M^t, M^{tu} \rangle$ a partial paraconsistent stable model. For every formula F constructed from propositional symbols, constant symbols representing consistent values in the underlying bilattice, and combinations of $\otimes_k$ and $\neg$ then $\widehat{M^p}(F) \leqslant_k \widehat{M^p}(-F)$.*

The corollary below is immediate, providing the confirmation that our semantics obeys Coherence:

Corollary 34. *All coherent answer sets and the paraconsistent well-founded model of a paraconsistent logic program P observe the Coherence Principle.*

Naturally, the Paraconsistent Well-founded Semantics extends Well-founded Semantics [24], and its generalizations *WFSX* [2,43] and *WFSX*$_p$ [3,4]. Similarly, Coherent Answer Sets extend the Answer Sets Semantics of Gelfond and Lifschitz [23]. For additional details, the reader is referred to [25].

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

Table 1
Decision table for *flu*

fever	*cough*	*headache*	*muscle_pain*	*flu*
no	no	no	no	no (in 99% of cases)
yes	no	no	no	no (in 80% of cases)
yes	yes	no	no	no (in 30% of cases)
yes	yes	no	no	yes (in 60% of cases)
yes	yes	yes	yes	yes (in 75% of cases)

5. Example

Paraconsistent Logic Programs have a large range of applications. We address the following example, adapted from [44], encoding a decision table based on rough relations. These are determined by *Rough Sets* (cf. [44]), introduced to deal with imprecise information. In [45,46], several logic programming languages are defined, making it possible to describe systems using rough relations and reason about them. On some points, these languages resemble our framework but they differ essentially in not enforcing coherence, and by limiting themselves to Belnap's four valued logic.

The symptoms *fever*, *cough*, *headache*, and *muscle_pain* are used to decide whether a patient has a flu. The diagnosis is performed according to decision table[2] of Table 1.

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

We resort to the PLP algebra $\mathfrak{R} = \langle [0,1]^2, i \rangle$ exhibited in Example 13 to encode the decision table. The first case is directly representable in paraconsistent logic programming by the rule:

$$\neg \mathit{flu} \leftarrow \langle 0.99, 0.0 \rangle \otimes_k \neg \mathit{fever} \otimes_k \neg \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle\_pain}. \tag{1}$$

Similarly, the second diagnosis case could be implemented via the next rule:

$$\neg \mathit{flu} \leftarrow \langle 0.8, 0.0 \rangle \otimes_k \mathit{fever} \otimes_k \neg \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle\_pain}. \tag{2}$$

The translation of the final case is immediate:

$$\mathit{flu} \leftarrow \langle 0.75, 0.0 \rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \mathit{headache} \otimes_k \mathit{muscle\_pain}. \tag{3}$$

Returning to rule (1), because of the way negation was defined (Definition 2) and since the truth-value of the head is greater or equal to that of the body in any model, one may alternatively construe it as:

$$\mathit{flu} \leftarrow \neg(\langle 0.99, 0.0 \rangle \otimes_k \neg \mathit{fever} \otimes_k \neg \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle\_pain})$$

which reduces to[3]

$$\mathit{flu} \leftarrow \langle 0.0, 0.99 \rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \mathit{headache} \otimes_k \mathit{muscle\_pain}. \tag{4}$$

[2] The figures are fictitious.

[3] From [7], we have $\neg(A \otimes_k B) = (\neg A \otimes_k \neg B)$.

To continue, the reader will surely notice that the body of rule (3) is identical to that of rule (4), with the exception of the confidence degrees in the two rules. According to WFM^p, the truth-value of *flu* is determined by the least upper bound (under $\leqslant_k$) of the truth-values assigned to the bodies of the rules for *flu*. Consequently, rules (1) (in its form (4)) and (3) can be combined into a single one:

$$flu \leftarrow \langle 0.75, 0.99\rangle \otimes_k fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain. \tag{5}$$

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, and *doesn't* have headache nor muscle-pain either, 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 supply negative evidence.

For the remaining situation (fever, cough, no headache, and no muscle_pain) two distinct rules are required for concluding whether the patient might or might not have a flu (as per cases (3) and (4)):

$$\neg flu \leftarrow \langle 0.3, 0.0\rangle \otimes_k fever \otimes_k cough \otimes_k \neg headache \otimes_k \neg muscle\_pain,$$
$$flu \leftarrow \langle 0.6, 0.0\rangle \otimes_k fever \otimes_k cough \otimes_k \neg headache \otimes_k \neg muscle\_pain.$$

So the paraconsistent logic program rules for diagnosing flu are:

$$\neg flu \leftarrow \langle 0.8, 0.0\rangle \otimes_k fever \otimes_k \neg cough \otimes_k \neg headache \otimes_k \neg muscle\_pain,$$
$$flu \leftarrow \langle 0.75, 0.99\rangle \otimes_k fever \otimes_k cough \otimes_k headache \otimes_k muscle\_pain,$$
$$\neg flu \leftarrow \langle 0.3, 0.0\rangle \otimes_k fever \otimes_k cough \otimes_k \neg headache \otimes_k \neg muscle\_pain,$$
$$flu \leftarrow \langle 0.6, 0.0\rangle \otimes_k fever \otimes_k cough \otimes_k \neg headache \otimes_k \neg muscle\_pain.$$

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

$$antibiotics \leftarrow \neg flu \tag{6}$$

or

$$antibiotics \leftarrow -\neg flu. \tag{7}$$

The former concludes 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[4] for *flu*.

[4] Which is not the same has having negative evidence!

Table 2
Models for the *flu* program

	fever	*cough*	*headache*	*muscle_pain*	*flu*	$\neg$*flu*	$-\neg$*flu*
T	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 0.99\rangle$	$\langle 0.99, 0.0\rangle$	$\langle 1.0, 0.01\rangle$
TU	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 0.99\rangle$	$\langle 0.99, 0.0\rangle$	$\langle 1.0, 0.01\rangle$
T	$\langle 1.0, 0.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 0.8\rangle$	$\langle 0.8, 0.0\rangle$	$\langle 1.0, 0.2\rangle$
TU	$\langle 1.0, 0.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 0.8\rangle$	$\langle 0.8, 0.0\rangle$	$\langle 1.0, 0.2\rangle$
T	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.6, 0.3\rangle$	$\langle 0.3, 0.6\rangle$	$\langle 0.4, 0.7\rangle$
TU	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.0, 1.0\rangle$	$\langle 0.6, 0.3\rangle$	$\langle 0.3, 0.6\rangle$	$\langle 0.4, 0.7\rangle$
T	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 0.75, 0.0\rangle$	$\langle 0.0, 0.75\rangle$	$\langle 0.25, 1.0\rangle$
TU	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 1.0, 0.0\rangle$	$\langle 0.75, 0.0\rangle$	$\langle 0.0, 0.75\rangle$	$\langle 0.25, 1.0\rangle$
T	$\langle 0.4, 0.6\rangle$	$\langle 0.7, 0.3\rangle$	$\langle 0.1, 0.9\rangle$	$\langle 0.2, 0.7\rangle$	$\langle 0.4, 0.3\rangle$	$\langle 0.3, 0.4\rangle$	$\langle 0.6, 0.7\rangle$
TU	$\langle 0.4, 0.6\rangle$	$\langle 0.7, 0.3\rangle$	$\langle 0.1, 0.9\rangle$	$\langle 0.2, 0.7\rangle$	$\langle 0.4, 0.3\rangle$	$\langle 0.3, 0.4\rangle$	$\langle 0.6, 0.7\rangle$
T	$\langle 0.4, 0.6\rangle$	$\langle 0.7, 0.3\rangle$	$\langle 0.7, 0.9\rangle$	$\langle 0.2, 0.7\rangle$	$\langle 0.4, 0.3\rangle$	$\langle 0.3, 0.4\rangle$	$\langle 0.7, 0.7\rangle$
TU	$\langle 0.4, 0.6\rangle$	$\langle 0.7, 0.3\rangle$	$\langle 0.1, 0.3\rangle$	$\langle 0.2, 0.7\rangle$	$\langle 0.3, 0.3\rangle$	$\langle 0.3, 0.3\rangle$	$\langle 0.6, 0.7\rangle$

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

The first model in Table 2 is obtained by adding the set of facts below to the previous rules, where the confidence degrees are extracted from the T component of the WFM^p model of the program:

$$\textit{fever} \leftarrow \langle 0.0, 1.0\rangle \qquad \textit{cough} \leftarrow \langle 0.0, 1.0\rangle$$
$$\textit{headache} \leftarrow \langle 0.0, 1.0\rangle \qquad \textit{muscle\_pain} \leftarrow \langle 0.0, 1.0\rangle$$

The first four models correspond to the (4) previously identified 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 in the columns for $\neg$*flu* and $-\neg$*flu*. By the Coherence Principle, it is always the case that, for every model, $\widehat{M^p}(\neg \textit{flu}) \leqslant_k \widehat{M^p}(-\neg \textit{flu})$, as the reader can check in the T-rows of each model. For instance, in the second model we have $\neg$*flu* with degree of evidence $\langle 0.8, 0.0\rangle$ and $-\neg$*flu* with evidence $\langle 1.0, 0.2\rangle$, and as expected $\langle 0.8, 0.0\rangle \leqslant_k \langle 1.0, 0.2\rangle$. Since flu has value $\langle 0.0, 0.8\rangle$, and $-\neg$*flu* has value $\langle 1.0, 0.2\rangle$, antibiotics should definitely be prescribed according to rule (7), confirming 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 then prescribed.

In the fifth model the physician is uncertain regarding each symptom. The interesting aspect of this case is that the degree of evidence for *flu* is obtained by combining together the degrees of evidence of several rules.

Finally, the last model illustrates how paraconsistency is handled by our semantics. The fact *headache* $\leftarrow \langle 0.7, 0.9\rangle$ is inconsistent in Fitting's sense (and thus in the WFM^p). In

the well-founded model of the program, *flu* has degree $\langle 0.3, 0.3\rangle$ in the TU-component, which is less (in the $\leqslant_k$ sense) than its degree of truth ($\langle 0.4, 0.3\rangle$). As we shall observe in the next section, this is indicative that *flu* is dependent on some inconsistent value: in our example, on the value assigned to *headache*. All the other previous models are consistent and fully-defined, i.e. they are coherent answer sets.

6. Capturing inconsistency and its dependencies

This section is devoted to the presentation of material and theorems related to inconsistencies and their propagation by the rules of a paraconsistent logic program. Specifically, we will provide a necessary and sufficient condition for inconsistency of a paraconsistent logic program; a necessary condition for dependence on inconsistency of a particular propositional symbol, and a counterexample of why this last condition is not also sufficient.

In order to motivate our results, note that in last model of Table 2, for two propositional symbols we have values whose TU-component $\leqslant_k$ T-component:

	headache	*flu*
T	$\langle 0.7, 0.9\rangle$	$\langle 0.4, 0.3\rangle$
TU	$\langle 0.1, 0.3\rangle$	$\langle 0.3, 0.3\rangle$

As we have mentioned, the T-component for *headache* is inconsistent in Fitting's sense (cf. Definition 4). The inconsistency of a propositional symbol is reflected in the paraconsistent well-founded model by forcing (via semi-normality) its TU-component to be less than its T-component according to knowledge order. However, although the T-value $\langle 0.4, 0.3\rangle$ assigned to *flu* is not in itself inconsistent, even so, *flu*, with its TU-value of $\langle 0.3, 0.3\rangle$, preserves this apparently bizarre behavior. In the sequel, we show such situations happen just when a propositional symbol depends on some inconsistency.

Before showing these results, we adopt some conventions motivated by the example forthwith:

Example 35. Given the PLP algebra $\mathfrak{R} = \langle [0,1]^2, i\rangle$, consider the programs P_1 and P_2:

$$P_1 = \begin{cases} a \leftarrow b \otimes_k c, \\ b \leftarrow \langle 0.6, 0.8\rangle, \\ c \leftarrow \langle 1.0, 0.0\rangle, \end{cases}$$
$$P_2 = a \leftarrow \langle 0.6, 0.8\rangle \otimes_k \langle 1.0, 0.0\rangle$$

whose respective paraconsistent well-founded models, M_1^p and M_2^p, indicate

$$M_1^t(a) = \langle 0.6, 0.0\rangle \quad \text{and} \quad M_1^{tu}(a) = \langle 0.4, 0.0\rangle,$$
$$M_2^t(a) = \langle 0.6, 0.0\rangle \quad \text{and} \quad M_2^{tu}(a) = \langle 0.6, 0.0\rangle.$$

We can explain why M_1^p and M_2^p disagree regarding "a" by noting that in P_1 we have an inconsistent constant being assigned to "b" through the rule $b \leftarrow \langle 0.6, 0.8\rangle$, whilst in P_2, we have no inconsistent value being assigned to a rule head since $\langle 0.6, 0.8\rangle \overset{\bullet}{\otimes}_k \langle 1.0, 0.0\rangle =$

$\langle 0.6, 0.0\rangle$. As consequence, in P_1, $M_1^t(b)$ differs from $M_1^{tu}(b)$, resulting therefore that $M_1^t(a)$ also differs from $M_1^{tu}(a)$. In contrast, in P_2, $M_2^t(a) = M_2^{tu}(a)$.

When proving the results below, the bilattice of truth-values are supposed to be *infinitary k-distributive*. We also assume a constant γ is only allowed to appear in rules of the form $\Phi[A] \leftarrow \gamma$. It is just a semantical restriction, because from a syntactical point of view, and without loss of generality, in a rule $\Phi[A] \leftarrow \Psi$ we can replace any constant γ appearing in Ψ by a new propositional symbol (say c'), and add the rule $c' \leftarrow \gamma$ to P. The purpose is to keep a tighter control on the propagation, via the program rules, of inconsistency generated by constants. Acting this way, we force inconsistent constants to behave as ordinary inconsistent propositional symbols. In the remaining part of this section, we treat constants and propositional symbols indistinctly.

We make use of Theorem 36 to ensure that, given a paraconsistent well-founded model $M^p = \langle M^t, M^{tu}\rangle$ for a program P, $M^t \sqsubseteq M^{tu}$ is equivalent to saying every propositional symbol in the language of P is consistent.

Theorem 36. *Let $M^p = \langle M^t, M^{tu}\rangle$ be the least fixed point of $\Gamma_P^{\mathfrak{B}}\Gamma_{P_s}^{\mathfrak{B}}$ for a paraconsistent logic program P. Then $M^t \sqsubseteq \Gamma_s^{\mathfrak{B}} M^t$ iff for every propositional symbol A in P language, $M^t(A) \leqslant_k \overset{\bullet}{-} M^t(A)$.*

Obviously, as it can be checked in the example below, if $M^t(A) \not\leqslant_k M^{tu}(A)$ there is some propositional symbol B inconsistent in M^p.

Example 37. Consider the paraconsistent logic program P showed in Example 35:

$$\begin{aligned}
\neg \mathit{flu} &\leftarrow \langle 0.8, 0.0\rangle \otimes_k \mathit{fever} \otimes_k \neg \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle_pain},\\
\mathit{flu} &\leftarrow \langle 0.75, 0.99\rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \mathit{headache} \otimes_k \mathit{muscle_pain},\\
\neg \mathit{flu} &\leftarrow \langle 0.3, 0.0\rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle_pain},\\
\mathit{flu} &\leftarrow \langle 0.6, 0.0\rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle_pain}.
\end{aligned}$$

Besides these rules, P contains the set of facts:

$$F_1 = \begin{cases} \mathit{fever} \leftarrow \langle 0.4, 0.6\rangle, & \mathit{cough} \leftarrow \langle 0.7, 0.3\rangle,\\ \mathit{headache} \leftarrow \langle 0.7, 0.9\rangle, & \mathit{muscle\_pain} \leftarrow \langle 0.2, 0.7\rangle. \end{cases}$$

The $WFM^p = \langle M^t, M^{tu}\rangle$ for P corresponds to the fifth model of Table 2, where we conclude $M^{tu}(\mathit{flu}) = \langle 0.3, 0.3\rangle <_k M^t(\mathit{flu}) = \langle 0.4, 0.3\rangle$. Accordingly, as ensured by Theorem 36, there is some propositional symbol (in this case, *headache*) inconsistent in the language of P, since $\langle 0.7, 0.9\rangle \not\leqslant_k \langle 0.1, 0.3\rangle = \overset{\bullet}{-}\langle 0.7, 0.9\rangle$.

Underpinned by Theorem 36, corollary below is immediate:

Corollary 38. *All coherent answer sets of a Paraconsistent Logic Program are consistent.*

In the next proposition we find out a bit more: whenever $M^t(A) \not\leqslant_k M^{tu}(A)$, either A is itself inconsistent or A depends on some inconsistency. However, before introducing it, let us motivate the notion of dependence:

Example 39. Let P be the program obtained by replacing the set of facts F_1 in Example 37 with the set F_2 below:

$$F_2 = \begin{cases} \mathit{fever} \leftarrow \langle 1, 0\rangle, & \mathit{cough} \leftarrow \langle 1, 0\rangle, \\ \mathit{headache} \leftarrow \langle 0, 1\rangle, & \mathit{muscle\_pain} \leftarrow \langle 0, 1\rangle. \end{cases}$$

The resultant $WFM^p = \langle M^t, M^{tu}\rangle$ for P corresponds to the third model of Table 2, where we conclude $M^t(\mathit{flu}) = M^{tu}(\mathit{flu}) = \langle 0.6, 0.3\rangle$. In the program P above, we have four rules for *flu*, but not all effectively contribute to the final value assigned to *flu* according to WFM^p. Clearly $M^t(\mathit{flu})$ and $M^{tu}(\mathit{flu})$ are obtained by combining the values assigned by the bodies of the rules

$$\neg \mathit{flu} \leftarrow \langle 0.3, 0.0\rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle\_pain},$$
$$\mathit{flu} \leftarrow \langle 0.6, 0.0\rangle \otimes_k \mathit{fever} \otimes_k \mathit{cough} \otimes_k \neg \mathit{headache} \otimes_k \neg \mathit{muscle\_pain}$$

present in the program of Example 37. When calculating the paraconsistent well-founded model for P, the values assigned by the first two rules are completely absorbed by the values assigned by the other two rules for *flu* mentioned in the example above. Consequently, without affecting the final result, we can eliminate the two first rules for *flu*.

In the proposition below, we just define the set U_A in order to gather all rules for $\Phi[A]$ which contribute to the value established for A according to WFM^p. The objective is to guarantee that if A is consistent, and for all rules r which effectively contribute to A ($r \in U_A$), if we have for all propositional symbols A' in the body of r, $M^t(A') \leqslant_k M^{tu}(A')$, then $M^t(A) \leqslant_k M^{tu}(A)$:

Proposition 40. *Let P be a paraconsistent logic program, A a propositional symbol of its language. Let $M^p = \langle M^t, M^{tu}\rangle$ be the paraconsistent well-founded model of P, and define*

$$\mathit{Body}_A = \left\{ \widehat{M^p}(\overline{\Phi}[\Psi]) \mid \Phi[A] \leftarrow \Psi \in P \right\} \quad \text{and}$$
$$U_A = \left\{ \Phi[A] \leftarrow \Psi \mid \widehat{M^p}(\overline{\Phi}[\Psi]) \text{ is an upper bound of } \mathit{Body}_A \right\}.$$

If the following conditions hold:

- $M^t(A) \leqslant_k \overset{\bullet}{-} M^t(A)$,
- *For every rule of the form $\Phi[A] \leftarrow \gamma$ in U_A, where γ is a constant symbol, we have $\overset{\bullet}{\gamma} \leqslant_k \overset{\bullet}{-}\overset{\bullet}{\gamma}$,*
- *For every rule $\Phi[A] \leftarrow \Psi[B_1, \ldots, B_m | C_1, \ldots, C_n]$ in U_A, we have $M^t(B_i) \leqslant_k M^{tu}(B_i)$ and $M^t(C_j) \leqslant_k M^{tu}(C_j)$,*

then $M^t(A) \leqslant_k M^{tu}(A)$.

In other words, if $M^t(A) \not\leqslant_k M^{tu}(A)$, then A is either inconsistent or there is some rule r for A such that for at least a propositional symbol (or constant) A' in the body of r, $M^t(A') \not\leqslant_k M^{tu}(A')$. Recalling Example 37, we have

$$M^{tu}(\mathit{headache}) = \langle 0.1, 0.3\rangle <_k M^t(\mathit{headache}) = \langle 0.7, 0.9\rangle,$$

and

$$M^{tu}(\mathit{flu}) = \langle 0.3, 0.3\rangle <_k M^t(\mathit{flu}) = \langle 0.4, 0.3\rangle.$$

Just by taking a look at the WFM^p for the program P we conclude, from Proposition 40, that *headache* is inconsistent and *flu* depends on an inconsistency (in this case, *headache*).

Because of the generality of the bilattice structure, we do not have the converse result for Proposition 40:

Example 41. Given the PLP algebra $\mathfrak{R} = \langle [0,1]^2, i\rangle$, let P be the paraconsistent logic program below:

$$P = \begin{cases} a \leftarrow b \otimes_k d \otimes_k -\neg a, & b \leftarrow \langle 0.7, 0.3\rangle, \\ c \leftarrow -\neg c, & d \leftarrow \langle 0.8, 0.3\rangle, \\ c \leftarrow e \otimes_k f, & e \leftarrow \langle 0.7, 0.6\rangle, \\ f \leftarrow \langle 0.6, 0.2\rangle, & g \leftarrow \langle 0.7, 0.7\rangle, \\ h \leftarrow g \otimes_k c, & i \leftarrow a \otimes_k h. \end{cases}$$

After evaluating the paraconsistent well-founded model $M^p = \langle M^t, M^{tu}\rangle$ for P, we obtain

$$\begin{aligned} M^t(i) &= \langle 0.3, 0.2\rangle, & M^{tu}(i) &= \langle 0.3, 0.2\rangle, \\ M^t(a) &= \langle 0.3, 0.3\rangle, & M^{tu}(a) &= \langle 0.7, 0.2\rangle, \\ M^t(h) &= \langle 0.6, 0.2\rangle, & M^{tu}(h) &= \langle 0.3, 0.3\rangle. \end{aligned}$$

Based on Proposition 40, both the propositional symbols "a" and "h" depended on inconsistency. In addition, given that the only rule for "i" in P is $i \leftarrow a \otimes_k h$, we can undoubtedly say "i" is also dependent syntactically on an inconsistency. However, as $M^t(a) \otimes_k M^t(h) = M^{tu}(a) \otimes_k M^{tu}(h)$, we have $M^t(i) = M^{tu}(i)$, and our clue for "i" depending on an inconsistency is lost! However, we note that "i" does not in this case actually depend on the inconsistent values for "a" and "h" to the point where these matter, for the value of "i" could be exactly the same for non-inconsistent values of the former. The open question is how to characterize and identify in a declarative and simple way such circumstances. We can still have the desired result in specific situations, when, for instance, we assume for every propositional symbol "a" in the program's language, $M^{tu}(a) \leqslant_k M^t(a)$ (a is strictly overdefined in M^p). As we can see in [47], this is enough to apply the converse result in Extended Logic Programs.

7. Conclusions, related work, and open issues

In this work we have fine-tuned some issues and explored new ones, arising from [47] in which we introduced Paraconsistent Logic Programs. These generalize Antitonic Logic

Programs [1] by ushering into an encompassing framework, of an appropriate kind, concepts to cope with explicit and default negation, and certifying that the latter complies with the Coherence Principle. Program rules are rather complex: heads are constituted by applications of unary functions to a propositional symbol admitting a reverse function with respect to $\leqslant_k$, whilst bodies correspond to compositions of arbitrary monotonic and antitonic operators, in each argument, over a complete bilattice in the sense of Ginsberg [5]. In order to define the semantics we then resort to a program division in the spirit of [22], which transforms paraconsistent logic programs into monotonic ones. Forthwith, we can similarly apply an immediate consequences operator, guaranteeing the existence of a minimum paraconsistent well-founded model WFM^p.

To motivate the use of Paraconsistent Logic Programs, and in order to map decision tables based on rough relations, we examine how our semantics manipulates inconsistency. Establishing some new results, we show how the semantics allows for paraconsistency at the same time it keeps, to some extent, a monitoring eye on the inconsistent information and its propagation through the program rules. As regards a complexity analysis of the WFM^p, at a first glance, it sounds expensive in terms of computational time, notwithstanding, for specific instances, that we can have inference methods based on polynomial algorithms, as in the implementation of $WFSX_p$ [4].

Indeed, in [47] we have provided a simple translation of Extended Logic Programs under $WFSX_p$ into Paraconsistent Logic Programs. Furthermore, other frameworks are easily embeddable into ours, such as Probabilistic Logic Programs [15], Possibilistic Logic Programming [11], Hybrid Probabilistic Logic Programs [13], Generalized Annotated Logic Programs [14], and Fuzzy Logic Programs [12]. On the one hand, these translations permit to simultaneously deal with negation, paraconsistency, and non-monotonic reasoning, within the uncertain formalisms above. On the other, we may now study the behavior of $WFSX_p$, and of other important paraconsistent semantics for Extended Logic Programs when uncertain reasoning, otherwise absent, is introduced into them.

Like Antitonic Logic Programs, we may envisage Paraconsistent Logic Programs as a natural extension of Fitting's works [7,27]. Particularly, our framework enables us to make a distinction between explicit negation and default negation, and to explore the coherence principle. In contradistinction to Fitting, we do not restrain our framework to interlaced bilattices. We should mention as well that the central point in these works is the logic programming syntax, instead of considering arbitrary isotonic and antitonic functions in the bodies. To be absolutely fair, the publication [27] introduces the notions of *attenuation operators*, which can be viewed as arbitrary monotonic operators over bilattices. In other related work [10], Arielli defines a semantics for extended logic programs also based on bilattices, but with a restricted syntax. The author too advocates the coherence principle in some specific situations, but it is not clear whether this is a general property of the semantics. We intend to explore the connections to this work in the future. In a similar direction, it is the work by Denecker et al. [9], but its focus is on the properties of the operators to guarantee the existence of a well-founded and stable model like semantics, while ours is in the definition of a language permitting the construction of such operators. Moreover, we rely on the combination of two related but distinct anti-monotonic operators.

The generality of our framework propels us to many possible future avenues of research: we may next generalize our structure to consider rules with more complex heads, for in-

stance, with disjunctions. In an opposite direction, we may explore particular instances of our framework, so as to improve the understanding of properties of concrete instances, and to compare them to existing work. Focusing on more theoretical results, we have detected promising links between our semantics and substructural logics [48,49], to be exposed in subsequent work. Based on these results, we shall define a model theory for our semantics in a style similar to that of Cabalar [50] and Pearce [51]. Given the close relationship between our semantics and $WFSX_p$, it is possible, by introducing a suitable operator to capture consistency, to obtain a model theory satisfying the *Gentle Principle of Explosion* [52]. Consequently, the resulting logic can be classified as a *Logic of Formal Inconsistency* (LFI) [52]. Another interesting line of research is the study of the various types of negation, specially if we allow for weak negation operators as well. This offers the opportunity for examining how the Coherence Principle functions in such cases. The definition of tabled derivation procedures is also envisaged, for some specific instances of the framework.

Acknowledgements

João Alcântara is a PhD student at UNL supported by CAPES, Brasília, Brazil. The authors thank the TARDE and FLUX projects, sponsored by FCT/MCES, Portugal. We also are grateful to Reinhard Kahle and José Alferes for helpful discussions. A specific word of appreciation should go to the anonymous referees for their detailed comments.

References

[1] C.V. Damásio, L.M. Pereira, Antitonic logic programs, in: T. Eiter, M. Truszczynski (Eds.), Proc. 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning, in: LNCS/LNAI, Springer, 2001, pp. 748–759.
[2] L.M. Pereira, J.J. Alferes, Well-founded semantics for logic programs with explicit negation, in: B. Neumann (Ed.), European Conference on Artificial Intelligence, John Wiley & Sons, Wien, Austria, 1992, pp. 102–106.
[3] J.J. Alferes, C.V. Damásio, L.M. Pereira, A logic programming system for non-monotonic reasoning, J. Automated Reas. (Special Issue) 14 (1) (1995) 93–147.
[4] C.V. Damásio, Paraconsistent extended logic programming with constraints, PhD thesis, Universidade Nova de Lisboa, 1996.
[5] M. Ginsberg, Multivalued logics: A uniform approach to reasoning in artificial intelligence, Comput. Intelligence 4 (1988) 265–316.
[6] M. Fitting, Bilattices in logic programming, in: G. Epstein (Ed.), 20th Internat. Symp. on Multiple-Valued Logic, IEEE, 1990, pp. 238–246.
[7] M. Fitting, Bilattices and the semantics of logic programming, J. Logic Programming 11 (1991) 91–116; URL citeseer.nj.nec.com/fitting89bilattices.html.
[8] M. Fitting, Fixpoint semantics for logic programming a survey, Theoret. Comput. Sci. 278 (1–2) (2002) 25–51.
[9] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: J. Minker (Ed.), Logic-Based Artificial Intelligence, Kluwer Academic Publishers, 2000, pp. 127–144.
[10] O. Arieli, Paraconsistent declarative semantics for extended logic programs, Ann. Math. Artificial Intelligence 36 (4) (2002) 381–417.
[11] D. Dubois, J. Lang, H. Prade, Towards possibilistic logic programming, in: Proc. ICLP'91, MIT Press, 1991, pp. 581–598.

[12] P. Vojtás, L. Paulík, Soundness and completeness of non-classical extended SLD-resolution, in: Proc. Ws. on Extensions of Logic Programming (ELP'96), in: LNCS, vol. 1050, Springer-Verlag, 1996, pp. 289–301.
[13] A. Dekhtyar, V.S. Subrahmanian, Hybrid probabilistic programs, J. Logic Programming 43 (3) (2000) 197–250; URL citeseer.nj.nec.com/dekhtyar00hybrid.html.
[14] M. Kifer, V.S. Subrahmanian, Theory of generalized annotated logic programming and its applications, J. Logic Programming 12 (1–4) (1992) 335–367; URL citeseer.nj.nec.com/kifer92theory.html.
[15] L. Lakshmanan, F. Sadri, On a theory of probabilistic deductive databases, Theory and Practice of Logic Programming 1 (1) (2001) 5–42.
[16] R. Ng, V.S. Subrahmanian, Probabilistic logic programming, INFCTRL: Information and Computation (formerly Information and Control) 101; URL citeseer.nj.nec.com/ng92probabilistic.html.
[17] E. Shapiro, Logic programs with uncertainties: A tool for implementing expert systems, in: Proc. IJCAI'83, William Kauffmann, 1983, pp. 529–532.
[18] M. van Emden, Quantitative deduction and its fixpoint theory, J. Logic Programming 3 (1) (1986) 37–53.
[19] M. Fitting, Logic programming on a topological bilattice, Fund. Math. XI (1988) 209–218.
[20] L. Lakshmanan, N. Shiri, A parametric approach to deductive databases with uncertainty, Knowledge Data Engrg. 13 (4) (2001) 554–570; URL citeseer.nj.nec.com/laks97parametric.html.
[21] C.V. Damásio, L.M. Pereira, Monotonic and residuated logic programs, in: S. Benferhat, P. Besnard (Eds.), 6th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, in: LNAI, vol. 2143, Springer, 2001.
[22] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K.A. Bowen (Eds.), 5th International Conference on Logic Programming, MIT Press, 1988, pp. 1070–1080.
[23] M. Gelfond, V. Lifschitz, Logic programs with classical negation, in: D.H.D. Warren, P. Szeredi (Eds.), 7th Int. Conf. on Logic Programming, MIT Press, 1990, pp. 579–597.
[24] A. van Gelder, K. Ross, J. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (3) (1991) 620–650.
[25] J. Alcântara, C.V. Damásio, L.M. Pereira, An encompassing framework for paraconsistent logic programs (extended version), Available at http://centria.di.fct.unl.pt/~cd/publicacoes/jalext03.ps.gz.
[26] N.D. Belnap, A useful four-valued logic, in: J.M. Dunn, G. Epstein (Eds.), Modern Uses of Multiple-Valued Logic, D. Reidel, 1977, pp. 8–37.
[27] M. Fitting, The family of stable models, J. Logic Programming 17 (1993) 197–225.
[28] J.J. Alferes, L.M. Pereira, T.C. Przymusinski, 'Classical' negation in nonmonotonic reasoning and logic programming, J. Automated Reas. 20 (1998) 107–142.
[29] R. Schweimeier, M. Schroeder, A parametrised hierarchy of argumentation semantics for extended logic programming and its application to the well-founded semantics, Theory and Practice of Logic Programming, submitted for publication.
[30] R. Kowalski, F. Sadri, Logic programs with exceptions, in: D.H.D. Warren, P. Szeredi (Eds.), 7th International Conference on Logic Programming, MIT Press, 1990.
[31] G. Wagner, A database needs two kinds of negation, in: B. Thalheim, J. Demetrovics, H.-D. Gerhardt (Eds.), Mathematical Foundations of Database Systems, in: LNCS, vol. 495, Springer-Verlag, 1991, pp. 357–371.
[32] D. Pearce, G. Wagner, Reasoning with negative information I: Strong negation in logic programs, in: L. Haaparanta, M. Kusch, I. Niiniluoto (Eds.), Language, Knowledge and Intentionality, in: Acta Philosophica Fennica, vol. 49, 1990, pp. 430–453.
[33] G. Wagner, Negation in fuzzy and possibilistic logic programs, in: Logic Programming and Soft Computing, Research Studies Press, 1998.
[34] J.H. Gallier, Logic for Computer Science, John Wiley & Sons, 1987.
[35] J.C. Mitchell, Foundations for Programing Languages, Foundations for Computing, MIT Press, 1996.
[36] G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, D. Scott, A Compendium of Continuous Lattices, Springer-Verlag, 1980.
[37] C.V. Damásio, L.M. Pereira, A survey of paraconsistent semantics for logic programs, in: D. Gabbay, P. Smets (Eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2, Kluwer Academic Publishers, 1998, pp. 241–320.
[38] C. Baral, V.S. Subrahmanian, Duality between alternative semantics of logic programs and nonmonotonic formalisms, J. Automated Reas. 10 (1993) 399–420.

[39] M. van Emden, R. Kowalski, The semantics of predicate logic as a programming language, J. ACM 4 (23) (1976) 733–742.
[40] L.M. Pereira, J.J. Alferes, J.N. Aparício, Default theory for well founded semantics with explicit negation, in: D. Pearce, G. Wagner (Eds.), Proc. European Workshop JELIA'92, in: LNAI, vol. 633, Springer-Verlag, 1992, pp. 339–356.
[41] A. Tarski, Lattice-theoretic fixpoint theorem and its applications, Pacific J. Math. 5 (1955) 285–309.
[42] H. Przymusinska, T.C. Przymusinski, Semantic issues in deductive databases and logic programs, in: R. Banerji (Ed.), Formal Techniques in Artificial Intelligence, a Sourcebook, North-Holland, 1990, pp. 321–367.
[43] J.J. Alferes, L.M. Pereira, in: Reasoning with Logic Programming, in: LNAI, vol. 1111, Springer-Verlag, 1996.
[44] Z. Pawlak, Rough Sets. Theoretical Aspects of Reasoning About Data, Kluwer Academic Publishers, Dordrecht, 1991.
[45] A. Vitória, C.V. Damásio, J. Małuszyński, Query answering in rough knowledge bases, in: 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC'2003), 2003, submitted for publication.
[46] A. Vitória, J. Małuszyński, A logic programming framework for rough sets, in: J. Alpigini, J. Peters, A. Skowron, N. Zhong (Eds.), Proc. of the 3rd International Conference on Rough Sets and Current Trends in Computing, RSCTC'02, in: LNCS/LNAI, vol. 2475, Springer-Verlag, 2002, pp. 205–212.
[47] J. Alcântara, C.V. Damásio, L.M. Pereira, Paraconsistent logic programs, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), 8th European Conf. on Logics in Artificial Intelligence, in: LNAI, vol. 2424, Springer-Verlag, 2002, pp. 345–356.
[48] R. Routley, V. Plumwood, R.K. Meyer, R.T. Brady, Relevant Logics and their Rivals, Ridgeview, 1982.
[49] G. Restall, An Introduction to Substructural Logics, Routledge, 2002.
[50] P. Cabalar, Well founded semantics as two-dimensional here-and-there, in: Answer Set Programming, AAAI Press, 2001, pp. 15–20 (AAAI Press, technical report SS-01-01), inside the 2001 AAAI Spring Symposium Series.
[51] D. Pearce, From here to there: stable negation in logic programming, in: D.M. Gabbay, H. Wansing (Eds.), What is Negation?, Kluwer Academic Publishers, 1999, pp. 161–181.
[52] W.A. Carnielli, J. Marcos, A taxonomy of **C**-systems, in: W.A. Carnielli, M.E. Coniglio, I.M.L. D'Ottaviano (Eds.), Paraconsistency: The Logical Way to the Inconsistent, Proceedings of the 2nd World Congress on Paraconsistency, held in Juquehy, Brazil, May 8–12, 2000, in: Lecture Notes in Pure and Applied Mathematics, vol. 228, Marcel Dekker, 2002, pp. 1–94.