On the Measure of Incoherence in Extended Residuated Logic Programs

Nicolás Madrid and Manuel Ojeda-Aciego

***Abstract*— In this paper we continue analyzing the introduction of negation into the framework of residuated logic programming [18], [19]; specifically, we focus on extended programs, in which strong negation is introduced. The classical approach to extended logic programs consists in considering negated literals as new, independent, ones and, then apply the usual monotonic approach (based on the fix-point semantics and the $T_{\mathbb{P}}$ operator); if the least fix-point so obtained is inconsistent, then the approach fails and no meaning is attached to the program. This paper introduces several approaches to considering consistence (under the term *coherence*) into a fuzzy setting, and studies some of their properties.**

I. Introduction

Inconsistency is usually an undesirable feature which arises naturally: for instance, consider a database consisting of various newspaper reports about one political event, it is hardly possible that the knowledge-base so obtained be consistent. Thus, it is advisable tolerating the inconsistency instead of rejecting it, see [6].

The problem of the formal management of inconsistency has been studied for more than three decades, and a number of different approaches have been introduced by researchers all over the world. There are practical reasons which suggest the development of formal frameworks for dealing with inconsistency, in [21] the authors argue that due to the lack of current software development methods in handling inconsistencies, software engineers have to resolve inconsistencies whenever they are detected, and this position sometimes generates adverse side-effects in the development process: When multiple conflicting solutions exist for the same problem, each solution should be preserved to allow further refinements along the development process. An early resolution of inconsistencies may result in loss of information and excessive restriction of the design space.

The first logical approaches can be dated back to the beginnings of the twentieth century with the development of paraconsistent logical systems, in any of its main orientations: for automated reasoning, for belief revision, for many-valued systems, for relevant systems, etc.

A paraconsistent approach for knowledge base integration allows keeping inconsistent information and reasoning in its presence, therefore it is not strange to find several approaches which follow this line: In [12] a paraconsistent logic is used as the underlying logic for the specification of P-Datalog, a deductive query language for databases containing inconsistent information; in [1] a framework is presented based on an arbitrary complete bilattice of truth-values, which allows a precise definition of strong and default negation. There are some recent approaches, which are particularly useful for non-monotonic reasoning and for drawing rational conclusions from incomplete and inconsistent information, such as [2] which introduces a general framework based on distance semantics and investigate the main properties of the induced entailment relations.

Nevertheless, the general interest on inconsistency-tolerant systems arose in the late eighties, in which there were some developments in the research line of deductive databases.

When integrating data coming from multiple different sources we are faced with the possibility of inconsistency in databases. There are many approaches directed to work with inconsistent knowledge-bases [11], [4], [9], [7], [16]. Most of them need at least three truth values {True,False,Inconsistent}. Therefore multi-valued logic and fuzzy logic seem to be useful frameworks to develop a inconsistent tolerance approach.

Other researchers have addressed the problem of managing inconsistent databases, i.e., databases violating integrity constraints, and propose a general logic framework for computing repairs and consistent answers over inconsistent databases, see for instance [3] or [15]. This paper is somewhat related to approaches of removing information from the knowledge-base that causes an inconsistency [5], [8]. However, working in a fuzzy framework allows us modifying the truth values of the formulas instead of removing them.

Our approach here is based on logic programming in a generalized context, namely, on residuated logic programming with strong negation. Therefore, our knowledge-bases have the form of extended residuated logic programs, that is, sets of IF-THEN rules with a literal in the head, with an arbitrary (finite) number of literals joined with a conjunctor, and weighted by values in a residuated lattice.

Inconsistence in this context arises when lifting the usual approach in classical logic to extended logic programs: negative literals are treated as new, independent, atoms and, then, the usual (monotonic) approach is applied. The least model of the program, if consistent, is accepted, otherwise, the semantics does not assign a meaning to the program. Our goal in this paper is to propose an adequate generalization of the concept of inconsistent interpretation in the realm of extended residuated logic programs.

This kind of programs is introduced in Section II. Then, in Section III we present a generalization of consistency in a multi-valued framework, which we have called *coherence* in order not to overlap other existing generalizations in the litera-

Nicolás Madrid and Manuel Ojeda-Aciego are with the School of Computer Science, University of Málaga, Blv Louis Pasteur s/n, E-29071, Málaga, Spain (phone: +34 952 132 871; email: {nmadrid,aciego}@ctima.uma.es).

ture. In the rest of the paper we focus on different measures of the incoherence of a extended residuated logic program. Our approach follows the four dimensions of inconsistency cited in [17]: atomic inconsistency, number of inconsistencies, size of inconsistency, degree of information.

II. Preliminary definitions

Definition 1: A *residuated lattice* is a tuple $(\mathcal{L}, \leq, *, \leftarrow)$ such that:

1) $(\mathcal{L}, \leq)$ is a complete bounded lattice, with top and bottom elements 1 and 0.
2) $(\mathcal{L}, *, 1)$ is a commutative monoid with unit element 1.
3) $(*, \leftarrow)$ forms an adjoint pair, i.e. $\forall x, y, z \in \mathcal{L}$

$$z \leq (x \leftarrow y) \quad \text{iff} \quad y * z \leq x$$

In residuated lattices one can interpret the operator $*$ like a conjunction and the operator $\leftarrow$ like an implication.

As usual a negation operator, over $\mathcal{L}$, is any decreasing mapping $n : \mathcal{L} \to \mathcal{L}$ satisfying $n(0) = 1$ and $n(1) = 0$. In the rest of the paper we will consider a residuated lattice enriched with a negation operator $\sim$, $(\mathcal{L}, \leq, *, \leftarrow, \sim)$. In order to introduce our logic programs, we will asume a set Π of propositional symbols. If $p \in \Pi$, then both p and $\sim p$ are called *literals*. Arbitrary literals will be denoted with the symbol ℓ (possible subscripted), and the set of all literals as Lit.

Definition 2: Given a residuated lattice with negation $(\mathcal{L}, *, \leftarrow, \sim)$, an *extended residuated logic program* $\mathbb{P}$ is a set of weighted rules of the form

$$\langle \ell \leftarrow \ell_1 * \cdots * \ell_m; \quad \vartheta \rangle$$

where ϑ is an element of $\mathcal{L}$ and $\ell, \ell_1, \ldots, \ell_n$ are literals.

Rules will be frequently denoted as $\langle \ell \leftarrow \mathcal{B}; \vartheta \rangle$. As usual, the formula $\mathcal{B}$ is called the *body* of the rule whereas ℓ is called its *head*. We consider *facts* as rules with empty body, which are interpreted as a rule $\langle \ell \leftarrow 1; \quad \vartheta \rangle$.

Definition 3: A fuzzy *$\mathcal{L}$-interpretation* is a mapping $I \colon Lit \to \mathcal{L}$; note that the domain of the interpretation can be lifted to any rule by homomorphic extension.

We say that I *satisfies* a rule $\langle \ell \leftarrow \mathcal{B}; \quad \vartheta \rangle$ if and only if $I(\mathcal{B}) * \vartheta \leq I(\ell)$ or, equivalently, $\vartheta \leq I(\ell \leftarrow \mathcal{B})$. Finally, I is a *model* of $\mathbb{P}$ if it satisfies all rules (and facts) in $\mathbb{P}$.

The set made up of every $\mathcal{L}$-interpretation is denoted by $\mathbb{I}_{\mathcal{L}}$. Usually we write interpretation instead of $\mathcal{L}$-interpretation and Program instead of Residuated logic Program.

III. Coherence and the semantics of extended residuated logic program

Let us start this section recalling the semantics for classical extended logic programs. In classical logic, the syntactic symbol $\sim$, occurring in extended logic programs, denote the strong negation, which is semantically different from default negation, usually represented by $\neg$. The semantics of $\neg$ is as follows: $\neg p$ is true if and only if p is not true, whereas $\sim p$ is true if and only if $\sim p$ can be inferred by the knowledge base. In other words, the truth value of p determines the truth value of $\neg p$ but does not determine the value of $\sim p$. Usually the two kinds of negations appear together in classical logic programs but in this paper we are only interested in programs with strong negations.

In the classical case, the semantics of extended programs is given by the least fix-point of the immediate consequence operator considering the negated literals as 'new' literals [14], when the obtained fix-point turns out to be inconsistent, then the program has no meaning. In fuzzy logic, the semantics is obtained in a similar way, iterating the immediate consequence operator defined in [18] for residuated logic programs. The crux of the matter now is when do we reject the obtained model. To answer this question the concept of consistence (or inconsistence) has to be generalized.

There are many ideas underlying the concept of inconsistence: conflicting inference, inferring contradiction formulas, lack of models, etc. We will focus our generalization in the idea of excess of information which leads to a conflict of the negation meta-rule. Let n be a negation operator, the negation meta-rule is defined as follow:*"if p has truth value ϑ then $n(p)$ has the truth value $n(\vartheta)$"*. The negation meta-rule is the idea underlying in the negation as failure and it has no use in computing the semantics for extended residuated logic programs, however it is crucial to determine when we reject models. Contradicting the negation meta-rule by excess of information means that the program rules infer more information for a propositional symbol p which could be inferred using the negation meta-rule.

The term *coherence* was considered in order to not overlap with other definitions of fuzzy-consistence in the literature. The notion of incoherent interpretation is given below:

Definition 4: A fuzzy L-interpretation I over Lit is *coherent* if the inequality $I(\sim p) \leq \sim I(p)$ holds for every propositional symbol p.

We have three main reasons to believe that incoherence is a good generalization of consistence in a fuzzy logic programming framework. Firstly, it is easy to implement because it only depends of the negation operator. Secondly, it allows lack of knowledge; for example, I such that $I(\ell) = 0$ for all $\ell \in Lit$ is always coherent. Finally, our notion of coherence coincides with consistence in the classical framework (it is easy to check that), with the important consequence that a coherent interpretation allows that two opposite literals, such as p and $\sim p$, live together ... under some requirements.

As we said above, given an extended residuated logic program $\mathbb{P}$ we can obtain its least model by considering the negated literals as new, independent, propositional symbols and then, iterating the immediate consequence operator. Obviously, the notion of coherence applies to programs as follows:

Definition 5: Let $\mathbb{P}$ be an extended residuated logic program, we say that $\mathbb{P}$ is *coherent* if its least model is coherent.

Although the definition of coherent program might look as a hard restriction, the following property of coherent interpretations shows that a program is coherent if and only if it has, at least, one coherent model.

Proposition 1: Let I and J be two interpretations satisfying $I \leq J$. If J is coherent, then I is coherent as well.

Corollary 1: An extended residuated logic program is coherent if and only if it has one coherent model.

In order to continue with some properties of the notion of coherence, take into account that an interpretation I assigns a truth degree to any negative literal $\sim p$ independently from the negation operator. This way, if we have two different negation operators ($\sim_1$ and $\sim_2$) we can talk about the coherence of I wrt any of these operators.

Proposition 2: Let $\sim_1$ and $\sim_2$ be two negation operators such that $\sim_1 \leq \sim_2$, then any interpretation I that is coherent wrt $\sim_1$ is coherent wrt $\sim_2$.

Example 1: Consider the lattice $[0,1]$ with the usual order, the Gödel connectives and the following program $\mathbb{P}$:

$$\begin{aligned} r_1 &: \langle p \leftarrow; \quad 1\rangle \\ r_2 &: \langle q \leftarrow p; \quad 0.8\rangle \\ r_3 &: \langle \sim q \leftarrow; \quad 0.7\rangle \end{aligned}$$

The least model is $M = \{(p,1);(q,0.8);(\sim q,0.7)\}$. If we consider the usual negation $n(x) = 1 - x$ to determine the coherence of the program we obtain that $\mathbb{P}$ is not coherent, and the least model semantics fails in this case. However, if we consider the negation:

$$\overline{n}(x) = \begin{cases} 1 & \text{if} \quad x \leq 0.8 \\ 0 & \text{if} \quad x \geq 0.8 \end{cases}$$

the program is coherent and the least model semantics provides a meaning to the program. □

As a consequence, note that the chosen negation operator to determine the coherence restricts, in some sense, the semantics of our programs.

We define an ordering among extended residuated logic programs as follows: Let $\mathbb{P}_1$ and $\mathbb{P}_2$ be two extended programs, then $\mathbb{P}_1 \subseteq \mathbb{P}_2$ if and only if for each rule $\langle r_i; \vartheta_1\rangle$ in $\mathbb{P}_1$ there exists another rule[1] $\langle r_i; \vartheta_2\rangle$ in $\mathbb{P}_2$ such that $\vartheta_1 \leq \vartheta_2$.

Proposition 3: Let $\mathbb{P}_1 \subseteq \mathbb{P}_2$ be two extended programs then the least model of $\mathbb{P}_1$ is smaller than the least model of $\mathbb{P}_2$.

Therefore we can say that the greater a program is the more information it provides. Now, as coherence represents excess of information, the following proposition holds easily:

Proposition 4: Let $\mathbb{P}_1 \subseteq \mathbb{P}_2$ be two extended programs. If $\mathbb{P}_2$ is coherent then $\mathbb{P}_1$ is coherent as well.

To finish this section, we give the definition of incoherent propositional symbol with respect to an extended residuated logic program:

Definition 6: Let $\mathbb{P}$ be a extended residuated logic program. Let $M_{\mathbb{P}}$ be the least model of $\mathbb{P}$, then a propositional symbol p is *incoherent* (wrt $\mathbb{P}$) if and only if $M_{\mathbb{P}}(\sim p) > \sim(M_{\mathbb{P}}(p))$.

In the rest of the paper we will keep using $M_{\mathbb{P}}$ to denote the least model of an extended residuated logic program $\mathbb{P}$ (either coherent or not).

[1]Note that the only difference between both rules is the assigned weight.

IV. Incoherence wrt propositional symbols

In this section, we will consider measuring coherence of a program in terms of the atomicity incoherence on the propositional symbols.

Let $\mathbb{P}$ be an extended residuated logic program, and recall that a propositional symbol $p \in \Pi$ is coherent wrt $\mathbb{P}$ iff p is coherent wrt $M_{\mathbb{P}}$, the least model of $\mathbb{P}$. If p is not coherent wrt $\mathbb{P}$ then p is incoherent wrt $\mathbb{P}$. Hereafter we will simply state *p is coherent/incoherent* without mentioning the program, whenever it is not ambiguous.

As stated in the introduction, a possibility to measure the amount of incoherence of a program is simply to count the number of incoherences, by determining the percentage of incoherent propositional symbols appearing in $\mathbb{P}$. We chose the latter.

We denote the number of incoherent propositional symbols appearing in $\mathbb{P}$ as $NI(\mathbb{P})$ (note that $NI(\mathbb{P})$ is a positive integer), and we define the incoherence measure I_1 as follows:

$$I_1(\mathbb{P}) = \frac{NI(\mathbb{P})}{|\Pi_{\mathbb{P}}|} \tag{1}$$

where $\Pi_{\mathbb{P}}$ denotes the crisp set of propositional symbols appearing in $\mathbb{P}$. Note that $I_1(\mathbb{P}) \in [0,1]$, if $I_1(\mathbb{P}) = 0$ then there are no incoherent propositional symbols in $\mathbb{P}$, that is, $\mathbb{P}$ is a coherent program. However, if $I_1(\mathbb{P}) = 1$ then every propositional symbol is incoherent in $\mathbb{P}$.

Note that the measure above just provides a notion of aggregated 'local incoherences'. It could be convenient to consider as well, some 'global' account of incoherences. For this purpose, we define a particular type of mappings in order to establish how incoherent an interpretation is.

A mapping $m\colon \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ is a pre-coherence measure if the following properties hold for all interpretation $I \in \mathbb{I}_{\mathcal{L}}$:

1) $m(I(\sim p), \sim I(p)) = 0$ iff p is a coherent propositional symbol wrt I.
2) m is monotonic wrt the first variable and antitonic wrt the second variable.

If $\mathcal{L}$ is a linear lattice then for each distance d defined over $\mathcal{L}$ we can make a coherence measure (called the coherence measure induced by the distance d) as follow:

$$m_d(x,y) = \begin{cases} 0 & \text{if} \quad x \leq y \\ d(x, \sim y) & \text{if} \quad x > y \end{cases}$$

The proof is easy and left to the reader.

Now, there are two ways to measure incoherent information: either estimating the maximal size of incoherence or estimating the average size of incoherence. For the former, we have the following definition:

$$I_2(\mathbb{P}) = \sup_{p \in \Pi_{\mathbb{P}}} \left\{ m\big(M_{\mathbb{P}}(\sim p), \sim M_{\mathbb{P}}(p)\big) \right\} \tag{2}$$

If there is a finite number of propositional symbols in the program, then the supremum is actually a maximum, and there exists $\overline{p}$ such that $I_2(\mathbb{P}) = m(M(\sim \overline{p}), \sim M(\overline{p}))$.

For the average size of incoherence, we need a greatest incoherent value for $\mathbb{P}$. Thanks to the second condition in the definition of pre-coherence measure, the value $m(\top, \bot)$ is the

maximal value which can be obtained by m. We define the third measure of incoherence as follows:

$$I_3(\mathbb{P}) = \frac{\sum_{p\in\mathbb{P}} m\big(M_{\mathbb{P}}(\sim p), \sim M_{\mathbb{P}}(p)\big)}{|\Pi_{\mathbb{P}}| \cdot m(\top, \bot)} \tag{3}$$

Note that I_3 is defined on two dimensions of incoherence: on the one hand, on the number of incoherent symbols and, on the other hand, provides a global amount of the incoherence in the program.

Proposition 5: For each $i = 1, 2, 3$, $\mathbb{P}$ is a coherent program if and only if $I_i(\mathbb{P}) = 0$.

Before studying the basic results of these measures, let us see an example which clarifies the different incoherence measures defined so far:

Example 2: Consider $\mathcal{L} = [0, 1]$, the pre-coherence measure induced by the Euclidean distance in $[0, 1]$, the product logic connectives and the negation operator $\sim(x) = 1-x$. Let $\mathbb{P}$ be the following program :

$$r_1 = \langle p \leftarrow, 1\rangle$$
$$r_2 = \langle q \leftarrow p, 0.9\rangle$$
$$r_3 = \langle \sim q \leftarrow p, 0.9\rangle$$
$$r_4 = \langle r \leftarrow \sim q, 0.8\rangle$$
$$r_5 = \langle \sim r \leftarrow p, 0.8\rangle$$

The minimal model of $\mathbb{P}$ is:

$$M_{\mathbb{P}} = \{(p, 1); (\sim p, 0); (q, 0.9); (\sim q, 0.9); (r, 0.72); (\sim r, 0.8)\}$$

In this example, we obtain the following measures for $\mathbb{P}$:

$$I_1(\mathbb{P}) = \frac{2}{3}$$
$$I_2(\mathbb{P}) = \max\{0, 0.8, 0.52\} = 0.8$$
$$I_3(\mathbb{P}) = \frac{1.32}{3} = 0.44$$

□

As stated above, incoherence represents just an excess of information. If this is so, when including new rules to a program, the incoherence measure should grow. The following proposition proves this fact.

Proposition 6: Let $\mathbb{P} \subseteq \mathbb{Q}$ be two extended residuated logic programs. Then:

$$I_i(\mathbb{P}) \leq I_i(\mathbb{Q}) \qquad \text{for all } i = 1, 2, 3;$$

Proof: Since $\mathbb{P} \subseteq \mathbb{Q}$, then $M_{\mathbb{P}} \leq M_{\mathbb{Q}}$, and the proof is trivial for $i = 1$.

For $i = 2$ and $i = 3$ the proof follows from the inequality

$$m(M_{\mathbb{Q}}(\sim p), \sim M_{\mathbb{Q}}(p)) \geq m(M_{\mathbb{P}}(\sim p), \sim M_{\mathbb{P}}(p))$$

for all $p \in \Pi_{\mathbb{P}}$. ■

The defined measures establish how incoherent a program is, but they say nothing about the reason of the incoherence. The main reason of the incoherence is the excess of information, and this information is generated by the rules. Therefore, the reason of the incoherence is in the rules. In the following sections, we define incoherence measures and information measures for rules with the aim of determining what rule/s can be safely removed in order to obtain a coherent program.

A. Rule-based incoherence

The aim here is to define other measures of incoherence for rules wrt an extended residuated logic program, which might help to determine a value of incoherence caused by a given rule in $\mathbb{P}$. These functions are based on the previous measures. We define for a rule r and an incoherent extended program $\mathbb{P}$ the following incoherence measures:

$$I_i(r; \mathbb{P}) = 1 - \frac{I_i(\mathbb{P} \smallsetminus \{r\})}{I_i(\mathbb{P})} \qquad i = 1, 2, 3$$

Where $\mathbb{P} \smallsetminus \{r\}$ denotes the program $\mathbb{P}$ without the rule r. Note that, as a consequence of Proposition 6, the value of these measures are in the unit interval: if $I_i(r; \mathbb{P}) = 0$, then rule r does not cause incoherence in $\mathbb{P}$, however, if $I_i(r; \mathbb{P}) = 1$ then we might obtain a coherent program by removing the rule r in $\mathbb{P}$.

B. On the information in a rule

In the previous section we have studied how incoherence can change when we remove a rule of a logic program. The aim of this section is to determine how many information is lost from the program when (some) rules are removed. We start by defining the information measure of a program.

Let $\mathbb{P}$ be an extended residuated logic program, then we define the information measure of $\mathbb{P}$ as follows:

$$\textsc{Inf}(\mathbb{P}) = \sum_{\ell \in Lit} m(M_{\mathbb{P}}(\ell), \bot)$$

The information measure is monotonic wrt the ordering between logic programs, that is:

Proposition 7: Let $\mathbb{P} \subseteq \mathbb{Q}$ be two extended residuated logic programs, then

$$\textsc{Inf}(\mathbb{P}) \leq \textsc{Inf}(\mathbb{Q}).$$

Given a rule r of $\mathbb{P}$, we can compute the amount of information lost when removing r from $\mathbb{P}$ as follows:

$$\textsc{Inf}(r; \mathbb{P}) = 1 - \frac{\textsc{Inf}(\mathbb{P} \smallsetminus \{r\})}{\textsc{Inf}(\mathbb{P})}$$

Observe that if $\textsc{Inf}(r; \mathbb{P}) = 0$ then r contributes no new information to $\mathbb{P}$.

Example 3: We continue with Example 2. The information measure of $\mathbb{P}$ is

$$\textsc{Inf}(\mathbb{P}) = 5.32$$

The results of the measures of incoherence and loss of information are shown in Figure 1.

Note that if we remove either rule r_1 or rule r_3, the new program is coherent. This occurs because $I_1(r_1, \mathbb{P}) = I_1(r_3, \mathbb{P}) = 1$. □

	$I_1(r_i;\mathbb{P})$	$I_2(r_i;\mathbb{P})$	$I_3(r_i;\mathbb{P})$	$\textsc{Inf}(r;\mathbb{P})$
r_1	1	1	1	1
r_2	0.5	0.35	0.61	0.15
r_3	1	1	1	0.3
r_4	0.5	0	0.4	0.13
r_5	0.5	0	0.4	0.15

Fig. 1. Results for Example 3.

V. Incoherence wrt rules

In this section we study coherence on the basis of sets of rules in the program. If we consider crisp sets of rules, then we could directly apply the crisp approaches based on set of formulae [5], [13] to our extended residuated logic programs.

However, our rules $\langle \ell \leftarrow \mathcal{B}; \vartheta\rangle$ differ from classical ones in an essential aspect, its weight. The value $\vartheta \in \mathcal{L}$ represents somehow the truth value of the rule in the program. In this way, we can think of modifying that value with the aim to obtaining a coherence measure.

For that purpose, we need to fix a t-norm $\mathcal{A}$ to handle the values of $\mathcal{L}$ (recall that a t-norm is a commutative and monotonic map $\mathcal{L}\times\mathcal{L}\to\mathcal{L}$ satisfying $\mathcal{A}(\bot,x)=\bot$ and $\mathcal{A}(\top,x)=x$). Fixed such t-norm, we can define a operator to modify the weights of rules.

Given an extended residuated logic program $\mathbb{P}$, a set $\{\langle r_i,\vartheta_i\rangle\}_i$ of rules in $\mathbb{P}$ and a value $\varphi\in\mathcal{L}$ we define a new extended residuated logic program $O_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i,\varphi)$ as follows:

$$O_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i,\varphi) = (\mathbb{P}\smallsetminus\{\langle r_i,\vartheta_i\rangle\}_i)\cup\{\langle r_i,\mathcal{A}(\vartheta_i,\varphi)\rangle\}_i$$

In other words, the operator $O_{\mathbb{P}}$ changes the weights of $\langle r_j,\vartheta_j\rangle\in\{\langle r_i,\vartheta_i\rangle\}_i$ by $\mathcal{A}(\vartheta_j,\varphi)$.

We define the coherence measure for a set of rules $\{\langle r_i,\vartheta_i\rangle\}_i\in\mathbb{P}$ as:

$$\begin{aligned}&\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i) = \\ &\quad = \sup\{\varphi\in\mathcal{L}\mid O_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i,\varphi) \textit{ is coherent}\}\end{aligned}$$

Note firstly that $\textsc{Coh}_{\mathbb{P}}$ cannot be defined for any set of rules; at end of this section we will return to this problem.

$\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i)$ somehow determines the degree to which the weights ϑ_i have to decrease in order to obtain a coherent program. If $\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i)=0$, then we can obtain a coherent program by removing the rules $\{\langle r_i,\vartheta_i\rangle\}_i$ in $\mathbb{P}$. However, if $\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i)=1$ we do not need to decrease the weights of $\{\langle r_i,\vartheta_i\rangle\}_i$ in order to obtain a coherent program.

Some properties of $\textsc{Coh}_{\mathbb{P}}$ are presented below:

Proposition 8: If $\mathbb{P}$ is a coherent extended residuated logic program then $\textsc{Coh}_{\mathbb{P}}(X)=\top$ for every set of rules $X\in\mathbb{P}$.

Proposition 9: $\mathbb{P}$ is a coherent residuated logic program if and only if $\textsc{Coh}_{\mathbb{P}}(X)=\top$ for a set of rules $X\in\mathbb{P}$.

Roughly speaking, Propositions 8 and 9 above say that a program is coherent if and only if we do not need to decrease the weights of any rule to obtain a coherent program.

The coherence measure $\textsc{Coh}_{\mathbb{P}}$ is antitonic wrt the order of extended residuated logic programs.

Proposition 10: Let $\mathbb{Q}\subseteq\mathbb{P}$ be two extended residuated logic programs, then for each set of rules $\{\langle r_i,\vartheta_i^{\mathbb{Q}}\rangle\}_i\in\mathbb{Q}$, there exists another set $\{\langle r_i,\vartheta_i^{\mathbb{P}}\rangle\}_i$ of rules in $\mathbb{P}$ such that:

$$\textsc{Coh}_{\mathbb{Q}}(\{\langle r_i,\vartheta_i^{\mathbb{Q}}\rangle\}_i)\geq\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i^{\mathbb{P}}\rangle\}_i)$$

Proof: As $\mathbb{Q}\subseteq\mathbb{P}$, then $\vartheta_i^{\mathbb{Q}}\leq\vartheta_i^{\mathbb{P}}$ for each $i\in\mathcal{I}$. Since the map $\mathcal{A}$ is monotonic then $\mathcal{A}(\vartheta_i^{\mathbb{Q}},\varphi)\leq\mathcal{A}(\vartheta_i^{\mathbb{P}},\varphi)$ and thus

$$O_{\mathbb{Q}}(\{\langle r_i,\vartheta_i^{\mathbb{Q}}\rangle\}_i,\varphi)\subseteq O_{\mathbb{P}}(\{\langle r_i,\vartheta_i^{\mathbb{P}}\rangle\}_i,\varphi)$$

Therefore, using Proposition 4, the inequality holds. ■

Proposition 10, in natural language, says us that the more information we have the more we should have to decrease the weights of the rules in order to recover coherence.

Proposition 11: The coherence measure $\textsc{Coh}_{\mathbb{P}}$ is monotonic, i.e if $\{\langle r_i,\vartheta_i\rangle\}_i\subseteq\{\langle r_j,\vartheta_j\rangle\}_j$ then $\textsc{Coh}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i)\leq\textsc{Coh}_{\mathbb{P}}(\{\langle r_j,\vartheta_j\rangle\}_{j\in\mathcal{J}})$.

Proof: As in Proposition 10, the proof follows from the fact that

$$O_{\mathbb{P}}(\{\langle r_i,\vartheta_i^{\mathbb{P}}\rangle\}_i,\varphi)\supseteq O_{\mathbb{P}}(\{\langle r_j,\vartheta_j^{\mathbb{P}}\rangle\}_{j\in\mathcal{J}},\varphi)$$

holds for all $\varphi\in\mathcal{L}$, together with Proposition 4. ■

A couple of examples should help to clarify the meaning of the coherence measure.

Example 4: We continue from Example 2. We consider the t-norm $\mathcal{A}=\min(x,y)$. The coherence measure for each single rule is:

	r_1	r_2	r_3	r_4	r_5
$\textsc{Coh}_{\mathbb{P}}(r_i)$	$5/9$	$-$	$1/9$	$-$	$-$

that is, we cannot obtain a coherent program removing one of the rules r_2, r_4 or r_5. However, if we decrease the weight of r_1, at least, to $5/9$ we obtain a coherent program. The same occurs if we decrease the weight of r_3 to $1/9$. □

Example 5: In some cases, removing single rules never provides a coherent model. For example, over $[0,1]$ and the Gödel connectives, we consider the following program:

$$r_1:\langle p\leftarrow;0.8\rangle$$

$$r_2:\langle q\leftarrow;0.8\rangle$$

$$r_3:\langle p\leftarrow q;0.8\rangle$$

$$r_4:\langle q\leftarrow p;0.8\rangle$$

$$r_5:\langle{\sim}p\leftarrow q;0.8\rangle$$

$$r_6:\langle{\sim}q\leftarrow p;0.8\rangle$$

The minimal model of this program is:

$$M_{\mathbb{P}}=\{(p,0.8);({\sim}p,0.8);(q,{\sim}0.8);({\sim}q,0.8)\}$$

If we consider the negation operator $n(x)=1-x$, then the program is not coherent and it makes sense measuring its incoherence. We start by measuring the coherence of the whole program $\mathbb{P}$ using the minimum t-norm:

$$\textsc{Coh}_{\mathbb{P}}(\mathbb{P})=0.5$$

That means that if we change the weight of every rule in the program by 0.5, we get a coherent program. However, if we change only the weight of a single rule, we never obtain a coherent program because $\text{COH}_{\mathbb{P}}$ does not get defined for any single rule. □

Let us consider again the problem of the existence of $\text{COH}_{\mathbb{P}}$ for a given set of rules. This is not a big problem, the inexistence of $\text{COH}_{\mathbb{P}}$ for the set of rules X simply means that we cannot obtain a coherent program removing X in $\mathbb{P}$. In some circumstances, the programmer could be interested in removing (or modifying) certain rules to obtain a coherent program, but $\text{COH}_{\mathbb{P}}$ is not defined for this set of rules. The solution is removing (or modifying) more rules than the (initially) desired. This solution is guaranteed by the following result.

Proposition 12: Let $\mathbb{P}$ be an extended residuated logic program. Then $\text{COH}_{\mathbb{P}}(\mathbb{P})$ is always defined.

Corollary 2: Let $\mathbb{P}$ be an extended residuated logic program. Then for each set X of rules in $\mathbb{P}$ there exists a set of rules Y such that $X \subseteq Y$ and $\text{COH}_{\mathbb{P}}(Y)$ is defined.

Remark 1: The inclusion operator in the above corollary is to be understood in the crisp sense. That is $X \subseteq Y$ means that every rule $\langle \ell \leftarrow \mathcal{B}; \vartheta \rangle$ in X belongs to Y as well (with the same weight).

Consider a set of rules $\{\langle r_i, \vartheta_i \rangle\}_i$ for which $\text{COH}_{\mathbb{P}}$ is undefined. Then, thanks to Corollary 2, we can consider minimal sets of rules containing it for which $\text{COH}_{\mathbb{P}}$ is defined. Thus, if we want to remove (or modify) the rules $\{\langle r_i, \vartheta_i \rangle\}_i$ to obtain a coherent program, we may modify by using $\text{COH}_{\mathbb{P}}$ any of these minimal sets.

Example 6: Continuing with Example 5. Suppose that the programmer wants to modify the rule r_6 to obtain a coherent program. Obviously $\text{COH}_{\mathbb{P}}(r_6)$ is not defined. However, r_6 is contained in some minimal set of rules in the sense of Corollary 2. The coherence measure for these minimal sets is as follows:

x	$\{r_6, r_5\}$	$\{r_6, r_1, r_2\}$	$\{r_6, r_1, r_3\}$	$\{r_6, r_2, r_4\}$
$\text{COH}_{\mathbb{P}}(x)$	0.2	0.5	0.2	0.2

VI. Conclusions and future work

A number of different measures for the incoherence of an extended residuated logic program have been presented. This is an important topic which deserves certain attention in order to provide adequate generalizations of the classical theory of classical logic programming with strong negation to fuzzy frameworks.

Much work still have to be done on this topic, in particular, to find possible connections with other existing approaches to inconsistent interpretations in the literature. This is an extremely wide area, since potentially interesting results may have been published in very different contexts. For instance, a potentially interesting research line for our goal seems to be that of the measures of contradiction between pairs of fuzzy sets [10].

References

[1] J. Alcântara, C. Damásio, and L. Pereira. An encompassing framework for paraconsistent logic programs. *Journal of Applied Logic*, 3(1):67–95, 2005.

[2] O. Arieli. Distance-based paraconsistent logics. *Int. J. Approx. Reasoning*, 48(3):766–783, 2008.

[3] P. Barceló and L. E. Bertossi. Logic programs for querying inconsistent databases. In *PADL '03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 208–222, London, UK, 2003. Springer-Verlag.

[4] N. D. Belnap. A Useful Four-Valued Logic. In J. M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel Pub., 1975.

[5] S. Benferhat, D. Dubois, and H. Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *In Proceedings of Uncertainty in Artificial Intelligence*, pages 411–419. Morgan Kaufmann, 1993.

[6] L. Bertossi, A. Hunter, and T. Schaub. Introduction to Inconsistency Tolerance. In A. H. Leopoldo Bertossi and T. Schaub, editors, *Inconsistency Tolerance*, Lecture Notes in Computer Science 3300, pages 1 – 14. Springer Verlag, 2005.

[7] P. Besnard and A. Hunter. Quasi-Classical Logic: Non-Trivializable Classical Reasoning From Inconsistent Information. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty 95*, Lecture Notes in Artificial Intelligence 946, pages 44–51. Springer Verlag, 1995.

[8] P. Besnard and A. Hunter. A Logic-based Theory of Deductive Arguments. *Artificial Intelligence*, 128:203–235, 2001.

[9] P. Besnard and T. H. Schaub. Signed Systems for Paraconsistent Reasoning. *Journal of Automated Reasoning*, 20:191–213, 1998.

[10] S. Cubillo, C. Torres, and E. Castiñeira. Self-contradiction and contradiction between two Atanassov's intuitionistic fuzzy sets. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 16(3):283–300, 2008.

[11] N. C. A. da Costa. On the Theory of Inconsistent Formal System. *Notre Dame Journal of Formal Logic*, 15 (4):497–510, 1974.

[12] S. de Amo and M. S. Pais. A paraconsistent logic programming approach for querying inconsistent databases. *Int. J. Approx. Reasoning*, 46(2):366–386, 2007.

[13] D. Dubois and H. Prade. Properties of measures of information in evidence and possibility theories. *Fuzzy Sets Syst.*, 100(supp.):35–49, 1999.

[14] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[15] G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. on Knowl. and Data Eng.*, 15(6):1389–1408, 2003.

[16] A. Hunter. Reasoning with Contradictory Information Using Quasi-Classical Logic. *Journal of Logic and Computation*, 10 (5):677–703, 2000.

[17] A. Hunter and S. Konieczny. Approaches to measuring inconsistent information. In *Inconsistency Tolerance. Volume 3300 of Lecture Notes in Computer Science*, pages 189–234. Springer, 2005.

[18] N. Madrid and M. Ojeda-Aciego. Towards a fuzzy answer set semantics for residuated logic programs. In *Proc of WI-IAT'08. Worshop on Fuzzy Logic in the Web*, pages 260–264, 2008.

[19] N. Madrid and M. Ojeda-Aciego. On coherence and consistence in fuzzy answer set semantics for residuated logic programs. *Lect. Notes in Computer Science*, 2009. To appear.

[20] Z. Majkic. Meta many-valued logic programming for incomplete and locally inconsistent databases. In *Proc of Database Engineering and Applications Symposium*, pages 459 – 461, 2004.

[21] F. Marcelloni and M. Aksit. Leaving inconsistency using fuzzy logic. *Information & Software Technology*, 43(12):725–741, 2001.