Measuring Inconsistency in Fuzzy Answer Set Semantics

Nicolás Madrid, *Student Member, IEEE* Manuel Ojeda-Aciego, *Senior Member, IEEE*

***Abstract*—Recent approaches have shown that the measurement of the amount of inconsistent information contained in a logic theory can be useful to infer positive information.**

This paper deals with the definition of measures of inconsistency in the residuated logic programming paradigm under the fuzzy answer set semantics. This fuzzy framework provides a soft mechanism of controlling the amount of information inferred and thus, controlling the inconsistencies by modifying slightly the truth-values of some rules.

***Index Terms*—Inconsistency, Fuzzy answer set semantics.**

I. Introduction

INCONSISTENCY has been considered for many years as an undesirable feature which has to be completely ignored in our logic theories. However inconsistency arises naturally in databases and, in many cases, seems to be unavoidable. For example, assume that a theft occurred in a classroom and we construct a knowledge-base containing the student's declarations; it is highly probable that we will obtain inconsistencies.

A typical reaction when somebody obtains an inconsistent knowledge-base is to reject it. Nevertheless, to reject the whole bulk of information provided by a knowledge-base is not a good decision, since we might be rejecting correct and useful information. In the "theft" example, if every student coincides with the time when the theft occurred, this data should be considered true in spite of the existence of contradictory information elsewhere.

Another way to deal with inconsistent knowledge-bases is to try to repair them. But it is worth taking into account that, in some cases, inconsistencies can provide us with useful information. Thus, it might be useful to develop some mechanism to tolerate inconsistent information instead of getting rid of it. For instance, if some students provide inconsistent declarations, then it is possible that they were involved in the pilfering; moreover, the *degree of inconsistency* of the declaration of a given student seems to be related to his/her being guilty. In this case, it would be desirable to have some means of measuring the amount of inconsistent information contained in a given logic theory.

Considering "*inconsistency-tolerant approaches*" [3] is not new, since the family of paraconsistent logics [33], introduced more than 30 years ago, allows us to handle efficiently inconsistent information. Among the paraconsistent approaches in the literature, we emphasize the approaches related to the *consistence restoring* [1], *fix-point semantics* [7] and *inconsistent information measuring* [6], [11], [42]. This paper is related to the latter approaches. Some other interesting works on measuring inconsistency in propositional knowledge bases are [11], [14]–[16], [19], [20].

The existence of a big number of paraconsistent approaches based on Belnap's Lattice [2], suggest that multi-valued and/or fuzzy logic is an ideal framework to handle inconsistency; for a survey on uncertainty and fuzzy LPs, you may refer to [37]. Our approach will be based on a particular type of fuzzy framework known as residuated logic programming [5]. Working in a fuzzy framework enables the possibility of restoring consistence by modifying slightly (it need not be either completely false or completely true) the information inferred of a given propositional symbol.

In this paper, we deal with measuring the amount of inconsistent information in general residuated logic programs under a convenient extension of the answer-set semantics [25]. Two different kinds of negations are included in the residuated logic programming framework: strong and default negation. The use of these two kinds of negation is advocated in many contexts of interest, particularly [41] justifies their use in relation to web rules.

Inconsistency is introduced as being composed of two different levels: "*lack of stable models*" (called instability) and "*incoherent stable models*" (called incoherence). The former occurs when a set of incompatible rules appears in the logic program, whereas the latter occurs when the existing models assign *contradictory* values to p and $\sim p$. It is important to point out that there is not a consensus on the concept of inconsistency in the fuzzy logic framework (see Section VIII).

Measures of inconsistency are defined paying attention to the reason which generates inconsistency [27]–[29]. In the case of incoherence, the measure is intended to represent the excess of information contained in the models; on the other hand, in the case of instability, the measure computes the minimal amount of information which has to be either removed or added to the program so that consistence is restored.

The paper is structured as follows: we start in Section II by recalling the syntax and semantics of general residuated logic programs; then, in Section III we introduce our extension of the fuzzy answer set semantics for this kind of programs. In Section IV we study the possible reasons which cause the inconsistency and, in addition, introduce the definition of measure of information that we will use later. In Sections V and VI we introduce, respectively, the measures of incoherence and instability since, as we stated above, these are the two dimensions into which inconsistency can be broken down.

N. Madrid and M. Ojeda-Aciego are with the Dept. Matemática Aplicada, ETSI Informática, Univ. de Málaga, Blv. Louis Pasteur 35, 29071 Málaga, Spain. (email: {nmadrid,aciego}@ctima.uma.es).

Later, in Section VIII we willcomment on related approaches to other measures of inconsistency. Finally, in Section IX we will draw the conclusions and describe some future work we will attempt in this research area. An appendix contains all the technical proofs of the results stated throughout the paper.

II. Preliminaries

Let us start this section by recalling the definition of residuated lattice, which fixes the set of truth-values and the relationship between the conjunction and the implication (the adjoint condition) occurring in residuated logic programs.

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

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

In residuated lattices, L represents the set of truth-values, the operator $*$ is interpreted as a conjunction and the operator $\leftarrow$ as an implication.

Hereafter, in the examples we will use the following connectives defined on the unit interval $[0, 1]$:

Gödel connectives

$$x *_G y = \min(x, y) \qquad x \leftarrow_G y = \begin{cases} 1 & \text{if} \quad x \geq y \\ x & \text{otherwise} \end{cases}$$

Product connectives

$$x *_P y = x \cdot y \qquad x \leftarrow_P y = \begin{cases} 1 & \text{if} \quad x \geq y \\ \frac{x}{y} & \text{otherwise} \end{cases}$$

Definition 2: A negation operator over a residuated lattice L is any decreasing operator $n \colon L \to L$ such that $n(0) = 1$ and $n(1) = 0$.

We will often use the following family of negation operators:

$$n_\alpha(x) = \begin{cases} 1 & \text{if} \quad x \leq \alpha \\ 0 & \text{otherwise} \end{cases}$$

In the rest of the paper we will consider a residuated lattice enriched with two negation operators, denoted by $\sim$ and $\neg$. Let us be more specific, $\sim$ denotes the strong negation whereas $\neg$ denotes the default negation. The difference between them is essentially semantic: the value of a propositional symbol negated by the strong negation has to be inferred directly by the program, whereas the value of a propositional symbol negated by the default negation depends on the value of the positive propositional symbol.

Remark 1: We will use the same symbol to denote the syntactic negation symbol and the negation operator associated with it whenever there is no possibility of confusion; otherwise, the symbols $\dot{\neg}$ and $\dot{\sim}$ will denote the negation operators associated to the respective negations.

Definition 3: A *literal* is either a propositional symbol or a propositional symbol negated by the strong negation $\sim$.

We denote arbitrary literals with the symbol ℓ (possibly subscripted) and the set of all literals as Lit. It is important to remark that propositional symbols negated by the default negation are no literals.

Definition 4: Given a residuated lattice with negations $(L, \leq, *, \leftarrow, \sim, \neg)$, a *general residuated logic program* $\mathbb{P}$ is a finite set of weighted rules of the form

$$\langle \ell \leftarrow \ell_1 * \cdots * \ell_m * \neg \ell_{m+1} * \cdots * \neg \ell_n; \quad \vartheta \rangle$$

where the weight ϑ is an element of L, and $\ell, \ell_1, \ldots, \ell_n$ are literals.

It is usual to denote the rules as $\langle \ell \leftarrow \mathcal{B}; \vartheta \rangle$. The formula $\mathcal{B}$ is usually called the *body* of the rule whereas ℓ is called its *head*. Sometimes, the body of a rule will be represented as consisting of two parts $\mathcal{B}^+$ and $\mathcal{B}^-$, where the former stands for $\ell_1 * \cdots * \ell_m$ and the latter for $\neg \ell_{m+1} * \cdots * \neg \ell_n$.

A *fact* is a rule with empty body, i.e facts are rules with this form $\langle p \leftarrow \quad ; \vartheta \rangle$. The set of literals symbols appearing in $\mathbb{P}$ is denoted by $Lit_{\mathbb{P}}$.

The following definition establishes a semantics for the syntax defined above.

Definition 5: An *L-valued interpretation* (for short, an L-interpretation) is a mapping $I \colon Lit \to L$. Note that the domain of the interpretation can be lifted to any rule by homomorphic extension.[1]

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 in $\mathbb{P}$.

Note that the order relation in the residuated lattice $(L, \leq)$ can be extended over the set of all L-interpretations as follows: *Let I and J be two L-interpretations, then $I \leq J$ if and only if $I(\ell) \leq J(\ell)$ for all literal $\ell \in Lit$.* Actually, the set of L-interpretations defines a complete lattice, where the top element is $I_\top(\ell) = 1$ for all $\ell \in Lit$ and the bottom element is $I_\bot(\ell) = 0$ for all $\ell \in Lit$. Note also that an L-interpretation can be interpreted as an L-fuzzy subset of the set of literals Lit.

Once the semantics for residuated logic programs has been defined, we can explain more carefully the semantic difference between default and strong negations. Let I be an L-interpretation; the truth-value of a propositional symbol negated by default, $\neg p$, with respect to (w.r.t.) I is obtained by applying the default operator $\dot{\neg}$ to the truth-value assigned to p by I. On the other hand, the truth-value of a strongly negated propositional symbol $\sim p$ w.r.t. I is assigned directly by the interpretation I. In other words, our approach is compositional w.r.t. default negation, but *needs not be* compositional w.r.t. strong negation.

In order to facilitate the description of fuzzy answer sets that will be given later, we classify the programs according to the kinds of negations appearing in them. A general residuated logic program $\mathbb{P}$ is said to be:

- *positive* or *definite* if it does not contain negation operators.
- *normal* if it does not contain strong negation but might contain default negation.
- *extended* if it does not contain default negation but it might contain strong negation.

[1] The value of one formula Φ is determined by the operators and the values assigned by I to each literal appearing in Φ. For example $I(\ell_1 \leftarrow \ell_2) = I(\ell_1) \leftarrow I(\ell_2)$.

III. Fuzzy Answer Set Semantics

Fuzzy answer sets will be incrementally defined by considering one type of negation at each step. Firstly, we will consider extended residuated logic programs, and we justify the introduction of the notion of coherence as a generalization to the fuzzy framework of the concept of consistence. Then, we define the fuzzy answer sets for general residuated logic programs by conveniently adapting the original definition of the *Gelfond-Lifschitz* reduct [10].

Remark 2: It is important to note that, for presentation purposes, we consider the residuated logic programming framework. However, the results can be straightforwardly extended to the multi-adjoint framework [31], [32] whose underlying structure is that of multi-adjoint lattice $(L, \leq, \{*_i, \leftarrow_i\}_i)$ where each $(L, \leq, *_i, \leftarrow_i)$ forms a residuated lattice. The language of the multi-adjoint framework is very flexible and general, and embeds those given, for instance, by [39] and other similar approaches [23], [24].

A. Extended Logic Programs and Coherence

As our interpretations are defined on the set of literals, we are able to extend the immediate consequence operator defined in [5] to extended residuated logic programs. The definition is essentially that of the positive case.

Definition 6: Let $\mathbb{P}$ be an extended residuated logic program. The immediate consequence operator maps every L-interpretation I to the L-interpretation $T_{\mathbb{P}}(I)$ defined below:

$$T_{\mathbb{P}}(I)(\ell) = \sup\{I(\mathcal{B}) * \vartheta \colon \langle \ell \leftarrow \mathcal{B}; \quad \vartheta \rangle \in \mathbb{P}\}$$

where $\ell \in Lit$.

$T_{\mathbb{P}}(I)$ represents the information which can be deduced immediately from $\mathbb{P}$ by considering the information in I as the knowledge base.

Proposition 1: The immediate consequence operator defined above is monotonic.

Now, by Knaster-Tarski's theorem we can state that $T_{\mathbb{P}}$ has a least fixpoint, denoted as $\mathrm{lfp}(T_{\mathbb{P}})$, which coincides with the least model of $\mathbb{P}$. As a result, we can obtain the least model of any extended residuated logic program by means of the least fixpoint of $T_{\mathbb{P}}$.

However, the semantics provided by $\mathrm{lfp}(T_{\mathbb{P}})$ does not take into account the interaction between opposite literals. In the classical case, the semantics given by the immediate consequence operator is rejected if the interpretation is inconsistent, i.e p and $\sim p$ are true at the same time. Thus we need to generalize the idea of inconsistency into the fuzzy framework.

The advantage of working in a fuzzy framework is that one can allow that two opposite literals, such as p and $\sim p$, live together ... under some requirements. As the domain of our interpretations is the set of literals, we can be more flexible and do not reject an interpretation contradicting the following rule of inference "(N) *if the value of one propositional symbol p is v, then the value of $\neg p$ is $n(v)$*" where n is a negation operator. For example, in classical logic, the interpretation assigning 0 to every literal contradicts the previous rule, but is not inconsistent.

Our generalization of inconsistency focuses on the fact that an acceptable interpretation cannot assign to each literal a value greater than the value obtained by using rule (N), since it would contain a contradiction with (N) by an excess of information. This idea can be described mathematically by using just the inequality $I(\sim p) \leq \dot{\sim} I(p)$. We have called *coherence* to our approach, in order to distinguish it from other existing definitions of consistence in a fuzzy setting.

Definition 7: An L-interpretation I over Lit is *coherent* if the inequality $I(\sim p) \leq \dot{\sim} I(p)$ holds for every propositional symbol p.

The notion of coherence has been studied in [26], [30] by providing motivations to consider this generalization of consistence instead of others. Apart from the fact that our notion of coherence coincides with consistence in the classical framework (it is easy to check that), there are three main reasons which support this definition as a good generalization of consistent interpretation in a fuzzy setting:

- Firstly, it is easy to implement since it only depends on the negation operator, contrariwise to other definitions which need to consider a t-norm as well.
- Secondly, it allows lack of knowledge. For example the interpretation $I_{\perp}$ which represents no information is always coherent.
- And thirdly, an incoherent interpretation implies a contradiction with the negation meta-rule by excess of information.

Remark 3: The following question might arise when the notion of coherence is introduced: why don't we consider the dual inequality $I(p) \leq \dot{\sim} I(\sim p)$? The answer is that, whenever the negation meta-rule holds then the corresponding interpretation always is coherent in the sense of Definition 7, as expected, whereas there exist negation operators for which the negation meta-rule holds and the dual inequality fails. As an example, consider, for instance, a negation operator $\sim$ such that $\sim(\sim x) < x$ for some $x \in L$ and an interpretation such that $I(p) = x$ and $I(\sim p) = \dot{\sim}(x)$; it is straightforward to see that $I(p) \leq \dot{\sim} I(\sim p)$ fails in this case.

Lack of coherence is an undesirable feature not only in the crisp case, but also in this extended case (see [30]). Therefore, we will be concerned mainly with *coherent* programs (those having at least one coherent model).

Proposition 2: The least model of an extended residuated logic program $\mathbb{P}$ is coherent if and only if $\mathbb{P}$ has (at least) one coherent model.

The following example shows the importance in the choice of the negation operator to establish when an extended residuated logic programs is coherent.

Example 1: Consider the following program $\mathbb{P}$ on the residuated lattice $([0,1], \leq, *_G, \leftarrow_G)$:

$$\begin{aligned} r_1 &: \langle p \leftarrow ; \quad 1.0 \rangle \\ r_2 &: \langle q \leftarrow p \quad 0.8 \rangle \\ r_3 &: \langle \sim q \leftarrow ; \quad 0.7 \rangle \end{aligned}$$

The least model of the program $\mathbb{P}$ is $M = \{(p,1); (\sim p, 0); (q, 0.8); (\sim q, 0.7)\}$. 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; however, if we consider the negation $n_{0.8}$ the program is coherent, and the least model semantics provides a meaning to the program. □

B. Fuzzy Answer sets

Once the concept of coherence has been presented, we introduce the notion of *fuzzy answer set* for general residuated logic programs. Such a set is a fuzzy set of literals, similarly to the classical case, which will be considered as a fuzzy L-interpretation. Our aim in this section is to adapt the approach given in [9] and [10] to the general residuated logic programs defined above.

To begin with, we have to define a transformation (the reduct construction) which transforms any general logic program into an extended logic program, modulo one interpretation.

In the classical case, the transformation comprises two actions: either remove a rule or remove default negative literals from the body of the rules. In our framework, the generalization of Gelfond-Lifschitz reduct is reduced to just one action, concerning elimination of default negation and re-computing the values of the weights of each rule.

Formally, let us consider a general residuated logic program $\mathbb{P}$ together with an L-interpretation I. We construct a new extended program $\mathbb{P}_I$ by substituting each rule in $\mathbb{P}$

$$\langle \ell \leftarrow \ell_1 * \cdots * \ell_m * \neg \ell_{m+1} * \cdots * \neg \ell_n; \quad \vartheta \rangle$$

by the rule[2]

$$\langle \ell \leftarrow \ell_1 * \cdots * \ell_m; \quad \dot{\neg} I(\ell_{m+1}) * \cdots * \dot{\neg} I(\ell_n) * \vartheta \rangle$$

Definition 8: The program $\mathbb{P}_I$ is called the *reduct* of $\mathbb{P}$ w.r.t. the interpretation I.

It is not difficult to prove that this definition generalizes the classical one. This is due to the fact that removing rules and substituting the weights by 0 are equivalent in the classical framework.

Remark 4: As a result of the definition, note that given two fuzzy L-interpretations I and J, then the reducts $\mathbb{P}_I$ and $\mathbb{P}_J$ have the same rules, and might only differ in the values of the weights. By the monotonicity properties of $*$ and $\neg$, we have that if $I \leq J$ then the weight of a rule in $\mathbb{P}_I$ is greater or equal than its weight in $\mathbb{P}_J$.

Notice that the new program $\mathbb{P}_I$ is extended, that is, it does not contain default negation. Therefore, we can consider the least model of $\mathbb{P}_I$ and provide the following definition.

Definition 9: An L-interpretation M is a *stable model* of a general residuated logic program $\mathbb{P}$ if and only if M is the least model of $\mathbb{P}_M$.

Lemma 1: Every stable model of $\mathbb{P}$ is a model of $\mathbb{P}$.

As stated by the previous result, the term *stable "model"* makes sense. In fact, stable models are minimal models, as stated by the following theorem:

Theorem 1: A stable model of $\mathbb{P}$ is a minimal model of $\mathbb{P}$.

Now, as in the classical case, the notion of stable model above can be used for defining *fuzzy answer sets* for general residuated logic programs.

[2]Note the overloaded use of the negation symbol, as a syntactic function in the formulas and as the algebraic negation in the truth-values.

Definition 10: Let $\mathbb{P}$ be a general residuated logic program. An L-interpretation I is said to be a *fuzzy answer set* of $\mathbb{P}$ iff it is a coherent stable model of $\mathbb{P}$.

The following example shows a logic program with only two rules in order to illustrate the notion of fuzzy answer set.

Example 2: Consider the general residuated logic program below:

$$\mathbb{P} = \{\langle p \leftarrow \neg \sim p; \vartheta \rangle \ ; \ \langle \sim p \leftarrow \neg p; \vartheta \rangle\}$$

where ϑ is an arbitrary element in $[0, 1]$, and the connectives are evaluated over $\{[0, 1], \leq, *_G, \leftarrow_G, \neg, \sim\}$ where the default negation is $\neg(x) = 1 - x$ and the strong negation is

$$\sim(x) = \begin{cases} 1 - x & \text{if} \quad 0.8 < x \leq 1 \\ \frac{3}{5} - \frac{1}{2}x & \text{if} \quad 0.4 < x \leq 0.8 \\ \frac{6}{5} - 2x & \text{if} \quad 0.2 < x \leq 0.4 \\ 1 - x & \text{if} \quad 0 \leq x \leq 0.2 \end{cases}$$

Let us check that if $\vartheta \leq 0.5$, then the program above has only one stable model M, exactly $M(p) = M(\sim p) = \vartheta$. Assume that M is a stable model, that is, it is the least model of $\mathbb{P}_M$ and has to coincide with the $[0, 1]$-interpretation $\{(p, \min(1 - M(\sim p), \vartheta)); (\sim p, \min(1 - M(p), \vartheta))\}$.

Now, since $\vartheta \leq 0.5$ then $M(p) \leq 0.5$ and $M(\sim p) \leq 0.5$; as a result we have that $1 - M(p) \geq 0.5 \geq \vartheta$ and $1 - M(\sim p) \geq 0.5 \geq \vartheta$. Therefore, $M(p) = \min(1 - M(\sim p), \vartheta) = \vartheta$. Similarly $M(\sim p) = \vartheta$.

On the other hand, let us check that if $\vartheta > 0.5$, then there exists a family of stable models given by the infinitely many interpretations below:

$$M_\delta \equiv \{(p, \delta); (\sim p, 1 - \delta)\} \text{ where } \delta \in [1 - \vartheta, \vartheta]$$

Firstly, we will see that every L-interpretation M_δ is a stable model of $\mathbb{P}$. The reduct $\mathbb{P}_{M_\delta}$ is the extended residuated logic program formed by two facts; namely $\langle p \leftarrow ; \min\{1 - M_\delta(\sim p), \vartheta\}\rangle$ and $\langle \sim p \leftarrow ; \ \min\{1 - M_\delta(p), \vartheta\}\rangle$. Since $M_\delta(p)$ and $M_\delta(\sim p)$ are greater or equal than $1 - \vartheta$, each fact in the reduct can be rewritten by $\langle p \leftarrow \ ; 1 - M_\delta(\sim p)\rangle$ and $\langle \sim p \leftarrow ; \ 1 - M_\delta(p)\rangle$ respectively. So the minimal model of $\mathbb{P}_{M_\delta}$ is the L-interpretation $I = \{(p, 1 - M_\delta(\sim p)), (\sim p, 1 - M_\delta(p))\} = \{(p, 1 - (1 - \delta)), (\sim p, 1 - \delta)\} = M_\delta$.

Let us see now that there is no other stable model, assume that M is a stable model such that $M(p) = \delta$. Then, as M is the least model of $\mathbb{P}_M$, the following equalities hold

$$\delta := M(p) = \min(1 - M(\sim p), \vartheta)$$

$$M(\sim p) = \min(1 - M(p), \vartheta) = \min(1 - \delta, \vartheta)$$

By using the second equality, we can assert that $M(\sim p)$ is equal either to ϑ or $1 - \delta$. We consider firstly the case $M(\sim p) = \vartheta$. So in this case $M(p) = \min(1 - \vartheta, \vartheta) = 1 - \vartheta$, since $\vartheta \geq 0.5$. So in this case, M is the stable model $M_{1-\vartheta} = \{(p, 1 - \vartheta); (\sim p, \vartheta)\}$. We consider now the case $M(\sim p) = 1 - \delta$. Then necessarily $1 - \delta \leq \vartheta$, or equivalently $\delta \geq 1 - \vartheta$. Moreover, as $M(p) = \min(1 - M(\sim p), \vartheta)$, $M(p) = \delta \leq \vartheta$. So in this case M is the stable model $M_\delta = \{(p, \delta); (\sim p, 1 - \delta)\}$ where $\delta \in [1 - \vartheta, \vartheta]$.

Note that not all of these stable models are fuzzy answer sets, since some of them are not coherent. For example, $M_{0.5} = \{(p, 0.5); (\sim p, 0.5)\}$ is a stable model of $\mathbb{P}$ for $\vartheta > 0.5$ but is not a fuzzy answer set (i.e is not coherent) since $I(\sim p) = 0.5 \not\leq \dot{\sim}(I(p)) = \dot{\sim}(0.5) = 0.35$.

More specifically,

- If $\vartheta \leq 0.2$ then the unique stable model is coherent and therefore is a fuzzy answer set as well
- If ϑ is in the open interval $(0.2, 0.8)$, then none of the existing stable models is a fuzzy answer set;
- Finally, if $\vartheta \geq 0.8$ the family of fuzzy answer sets is:

$$M_\delta \equiv \{(p, \delta); (\sim p, 1-\delta)\} : 1 - \vartheta \leq \delta \leq \vartheta, \\ \delta \in (0, 0.2) \cup (0.8, 1)\} \quad \square$$

Notice that the weights of the rules are crucial to determine if a program has, or has not, a fuzzy answer set.

We can now extend the definition of *inconsistent* program to the framework of residuated logic programming as follows:

Definition 11: A residuated logic program $\mathbb{P}$ is said to be *inconsistent* if there is no fuzzy answer set of $\mathbb{P}$. Otherwise $\mathbb{P}$ is said to be *consistent*.

In the following section we describe reasons which can cause inconsistency in residuated logic programs.

IV. Causes of Inconsistence: Information Measure

From a technical point of view, inconsistency of a residuated logic program $\mathbb{P}$ can be due to one of the following two reasons:

- **Incoherence**: Every interpretation such that $I = \mathrm{lfp}(\mathbb{P}_I)$ is incoherent.
- **Instability**: There are no interpretations I satisfying that $I = \mathrm{lfp}(\mathbb{P}_I)$.

This technical distinction enables the definition of measures focusing on each cause of inconsistency. This way, we can classify inconsistent logic programs into two sets:

Definition 12: Let $\mathbb{P}$ be a general residuated logic program. It is said that:

- $\mathbb{P}$ is *unstable* if and only if $\mathbb{P}$ has not a stable model. Otherwise $\mathbb{P}$ is called *stable*.
- $\mathbb{P}$ is *incoherent* if $\mathbb{P}$ is stable and every stable model of $\mathbb{P}$ is incoherent.

The main difference between both features is that incoherence is intrinsic to interpretations whereas the instability is not: In section V we will measure the level of inconsistency of a stable program by measuring the inherent incoherence in each stable model. On the other hand, measuring the inconsistency caused by instability requires a deeper study on the causes of inconsistency, since no stable model exists.

When representing knowledge as a logic program it is usual to implement rules according to a set of external data (obtained either from sensors or at the suggestion of an expert); this data can be subject to errors and/or imprecision, and may lead to the following shortcomings:

- Not including relevant information. (Missing information)
- Including information which is either false or leading to contradiction. (Excess of information)

Any of the situations above might lead to inconsistency. Let us further discuss this by means of an example: assume that in a program to encode the protocols to treat physical abuse we include the following rules to describe some psychological emotions:

$$\begin{aligned} r_1 &: \langle Fear \leftarrow \neg Agressive *_G TenseSituation \quad ; 0.6\rangle \\ r_2 &: \langle Relax \leftarrow \neg Fear \quad ; 0.6\rangle \\ r_3 &: \langle Excited \leftarrow \neg Relax *_G TenseSituation \quad ; 0.6\rangle \\ r_4 &: \langle Agressive \leftarrow Excited *_G Abuser \quad ; 0.6\rangle \end{aligned}$$

where the negation operator associated with $\neg$ is $n_{0.4}$.

The first rule determines that if we do not know that a person is aggressive and she is in a dangerous situation then she fells fear (the interpretation of the other three rules is similar). These four rules do not imply any contradiction, in fact, the program consisting of the four rules above has just one stable model $I = \{(Relax, 0.6)\}$. However, if we add the following facts

$$\begin{aligned} r_5 &: \langle TenseSituation \leftarrow \quad ; 0.8\rangle \\ r_6 &: \langle Abuser \leftarrow \quad ; 0.7\rangle \end{aligned}$$

the program turns out to be unstable. What are the reasons for this behaviour?

As stated above, it may be because of excess or lack of information. For the former, excess of information can reside in any subset of rules (either singleton or not), it might be that too much information is obtained by default from r_1, r_2 and r_3. Notice that if the weights of these rules are changed by a value smaller than 0.4, therefore reducing the amount of information provided by those rules, the program would remain stable.

Lack of information is more difficult to handle, since we do not know which rules are missing. In principle, there are three possibilities by which to recover the missing information:

- Adding facts. That is, include positive information about propositional symbols which can be inferred from real-world observation. For example, if we include the fact $\langle Fear \leftarrow \quad ; 0.5\rangle$, the program gets stable again.
- Adding proper rules. In this case, the new rules permit us to draw consequences which allow for recovering stability. For example, if we include the rule

$$r : \langle Aggressive \leftarrow Fear *_G Abuser *_G \\ *_G TenseSituation \quad ; 0.6\rangle$$

  then, the program gets stable again.
- Adding hypotheses to the body of some rules. This case may occur when the program has been built from observable data in a fixed context, and some information was not considered relevant in a first approach. Continuing with the previous example, it is possible that rule r_2 represents a default reasoning done in non-dangerous situation, where the usual behavior is to be relaxed. However in tense situations, relax could not be a correct default inference. As a result, the missing information in the previous example might be due to not considering, for instance, a fact like $\neg(TenseSituation)$ in the body

of r_2; notice that if we do that, the program gets stable again.

Our approach to measure the inconsistency of a residuated logic program is divided according to its source. Firstly, assuming that stable models exist, by measuring the incoherence of these stable models. Secondly, if no stable model exists, we will measure the inconsistency by means of the minimum amount of information which we have either to remove or to add in order to obtain a stable logic program.

Removing or adding information in a residuated program can be done essentially by modifying the weights of the rules and facts, since the lesser (resp. bigger) they are, the less (resp. more) information is produced. The key point is how to measure the amount of information which has to be either removed or added.

We propose to assign to each element in the lattice a value corresponding to the inherent information it contains. That is, we propose to fix an operator $m\colon L \to \mathbb{R}^+$ such that:

- $m(x) = 0$ if and only if $x = 0$
- m is monotonic
- $m(\sup(x, y)) \geq m(x) + m(y) - m(\inf(x, y))$ for all $x, y \in L$

The third condition is required since the information contained in the supremum of two elements should reflect the amount of information contained in each element separately, therefore $m(\sup(x, y))$ should be greater than $m(x)+m(y)-m(\inf(x, y))$, since the latter part is counted twice when adding $m(x)$ and $m(y)$.

Such an operator will be called an *information measure*. Note that the third item does not impose any restriction if the lattice is linear. It is not difficult to define this kind of operators in a lattice:

Example 3: Any norm $||\cdot||$ on the lattice $([0, 1], \leq)$ is an information measure, since $||x|| = 0$ if and only if $x = 0$; and if $x \leq y$ then

$$||x|| = ||\frac{x}{y} \cdot y|| = |\frac{x}{y}| \cdot ||y|| \leq ||y|| \qquad \square$$

Example 4: Let $(L, \leq)$ be a finite lattice. An information measure can be defined as follows:

$$m(a) = \begin{cases} 0 & \text{if } x = 0 \\ 1 + \sum_{x \leq a} m(x) & \text{otherwise} \end{cases}$$

Clearly it defines an information measure. $\square$

From now on, we will consider that our underlying lattices of truth-values have an associated information measure.

V. Measuring Incoherence in Stable Residuated Logic Programs

We start this section by defining measures of incoherence directly over interpretations in order to use them on stable residuated logic program later.

A. Measures of Incoherence on Interpretations for a Program

A first possibility is to consider the ratio of incoherent propositional symbols in an L-interpretation with respect to the number of propositional symbols occurring in program $\mathbb{P}$.

To begin with, let us particularize the notion of coherence so that it can be applied to a single propositional symbol.

Definition 13: A propositional symbol p is coherent w.r.t. an L-interpretation I if and only if $I(\sim p) \leq \dot{\sim} I(p)$. A propositional symbol is incoherent w.r.t. I if is not coherent w.r.t. I.

Definition 14: Let I be an L-interpretation. We define the measure of incoherence

$$\mathcal{I}_1(I) = \frac{\mathcal{NI}(I)}{|\Pi_{\mathbb{P}}|} \tag{1}$$

where $\mathcal{NI}(I)$ denotes the number of incoherent propositional symbols in $\mathbb{P}$, and $\Pi_{\mathbb{P}}$ denotes the set of propositional symbols occurring in $\mathbb{P}$.

Note that $\mathcal{I}_1(I) \in [0, 1]$; if $\mathcal{I}_1(I) = 1$ then every propositional symbol in the domain of I is incoherent, whereas if $\mathcal{I}_1(I) = 0$, then there are no incoherent propositional symbols in $\mathbb{P}$ w.r.t. I.

The measure above provides just a notion of aggregated *"local incoherences"*; however, one should also take into account the *amount* of violation of the condition of coherence. For instance, with the negation operator $\sim(x) = 1 - x$, the propositional symbol p is incoherent w.r.t. the two interpretations I_1 and I_2 below:

$$\begin{array}{ll} I_1(p) = 0.5 & I_2(p) = 0.9 \\ I_1(\sim p) = 0.6 & I_2(\sim p) = 1 \end{array}$$

Certainly, I_2 seems more incoherent than I_1 since I_2 breaks the coherence condition in a "bigger degree" than I_1.

In order to define an incoherence measure which takes into account this "degree" of incoherence for propositional symbols, we define the set of *coherent pairs* of a propositional symbol p w.r.t. a negation operator, $\sim$, and an L-interpretation I, as follows:

$$\Delta_I^{\sim}(p) = \{(x, y) \in L \times L\colon\ x \leq I(p),\ y \leq I(\sim p),\ y \leq \dot{\sim}(x)\}$$

The set $\Delta_I^{\sim}(p)$ consists of all possible pairs of values representing coherent information for p (and $\sim p$) in terms of I. The definition of $\Delta_I^{\sim}$ and the information measure m associated to $(L, \leq, *, \leftarrow)$ allow us to provide the following measure of incoherence for a propositional symbol p w.r.t. an L-interpretation I:

$$\mathcal{I}(p; I) = \inf_{(x,y) \in \Delta_I^{\sim}(p)} \{m(I(p)) - m(x) + m(I(\sim p)) - m(y)\}$$

Note that $\mathcal{I}(p; I) \geq 0$ since $m(I(p)) \geq m(x)$ and $m(I(\sim p)) \geq m(y)$ for all $(x, y) \in \Delta_I^{\sim}(p)$. The idea underlying the definition of $\mathcal{I}(p; I)$ is to determine how much information has to be removed at least from $I(p)$ and $I(\sim p)$ in order to recover coherence. If we remove a certain amount of information α from $I(p)$, and a certain amount of information β from $I(\sim p)$, then we actually modify information amounting $\alpha + \beta$. Therefore, the measure $\mathcal{I}(p; I)$ determines in some sense the least amount of information that has to be removed in order to obtain coherent information about p w.r.t. I.

Example 5: For the interpretations I_1 and I_2 given above, we obtain $\mathcal{I}(p; I_1) = 0.1$ and $\mathcal{I}(p; I_2) = 0.9$ by considering the information measure induced by the Euclidean norm.

Remark 5: The definition of $\mathcal{I}(p;I)$ collapses to an extremely simple and intuitive form in the specific case of the unit interval ($L=[0,1]$), the information measure induced by the Euclidean norm, and $\sim(x)=1-x$:

$$\mathcal{I}(p;I)=\begin{cases} 0 & \text{if } p \text{ is coherent w.r.t. } I \\ I(\sim p)-\dot{\sim}I(p) & \text{otherwise} \end{cases}$$

We focus below on some properties of $\mathcal{I}(p;I)$:

Proposition 3: If the propositional symbol p is coherent w.r.t. I then $\mathcal{I}(p;I)=0$.

The converse is not true in general.

Example 6: Consider the lattice $[0,1]$ with the information measure induced by the Euclidean norm and the following negation operator

$$\dot{\sim}(x)=\begin{cases} 0 & \text{if } x\geq 0.5 \\ 1 & \text{if } x<0.5 \end{cases}$$

Then the propositional symbol p w.r.t. the interpretation $I(p)=0.5$, $I(\sim p)=0.5$ is incoherent and $\mathcal{I}(p;I)=0$.

Although the equivalence between null measure of incoherence and coherence is not true in general, there exist frameworks in which both notions coincide:

Proposition 4: Assume that the residuated lattice is finite and the information measure used is injective. Then, if $\mathcal{I}(p;I)=0$, the propositional symbol p is coherent w.r.t. I.

Proposition 5: On the unit interval $[0,1]$ and under a continuous and injective information measure; if $\dot{\sim}$ (the operator associated to strong negation) is continuous, then $\mathcal{I}(p;I)=0$ iff p is coherent w.r.t. I.

The following result relates the ordering between L-interpretations and the measure of incoherence for propositional symbols: the greater is an L-interpretation more incoherent is.

Proposition 6: Let $I\leq J$ be two L-interpretations. Then $\mathcal{I}(p;I)\leq\mathcal{I}(p;J)$ for all propositional symbol p.

The following proposition shows that $\mathcal{I}(p;I)$ is bounded by the inherent information in $I(p)$:

Proposition 7: Let I be an L-interpretation, then

$$\mathcal{I}(p;I)\leq\min\left\{m(I(p)),m(I(\sim p))\right\}$$

As a consequence of the proposition above, $m(1)$ is actually an upper bound for the value of each $\mathcal{I}(p;I)$.

Now, there are two ways to measure incoherent information in an L-interpretation: either estimating the maximum size of incoherence, or estimating the average size of incoherence. For the former, we have the following definition:

$$\mathcal{I}_2(I)=\max_{p\in\Pi_{\mathbb{P}}}\{\mathcal{I}(p;I)\} \tag{2}$$

For the average size of incoherence we consider:

$$\mathcal{I}_3(I)=\frac{\sum_{p\in\Pi_{\mathbb{P}}}\mathcal{I}(p;I)}{|\Pi_{\mathbb{P}}|\cdot\mathcal{I}(p;I_\top)} \tag{3}$$

As consequence of Proposition 3 we have that if I is coherent, then $\mathcal{I}_i(I)=0$ for all $i=1,2,3$.

We provide below some examples to illustrate these measures of incoherence.

Example 7: Consider the unit interval and the interpretation given by the following table:

x	$I(x)$	$I(\sim x)$
p	0.7	0.7
q	0.7	0.5
r	0.2	0.8
s	0.7	0
t	1	0

If we use the Euclidean measure and the negation operator $\sim(x)=1-x$ then the incoherence measure of each propositional symbol are given by the following table:

x	p	q	r	s	t
$\mathcal{I}(x;I)$	0.4	0.2	0	0	0

Hence, we can measure the incoherence of I by using the values above:

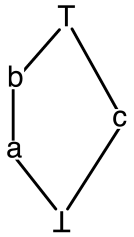
$$\mathcal{I}_1(I)=\frac{2}{5}=0.4$$

$$\mathcal{I}_2(I)=\max\{0.4,0.2,0,0,0\}=0.4$$

$$\mathcal{I}_3(I)=\frac{0.4+0.2+0+0+0}{5\cdot 1}=\frac{0.6}{5}=0.12 \qquad \square$$

The example below uses a non-linear lattice.

Example 8: Consider the lattice and information measure given by:



x	$m(x)$
$\bot$	0
a	1
b	3
c	2
$\top$	4

together with the L-interpretation and negation below

x	$I(x)$	$I(\sim x)$
p	b	c
q	$\top$	b
r	a	$\bot$

x	$\sim(x)$
$\bot$	$\top$
a	b
b	a
$\top$	$\bot$

It is not difficult to obtain the following measures of incoherence for I:

$$\mathcal{I}_1(I)=\frac{2}{3} \quad ; \quad \mathcal{I}_2(I)=\max\{2,3,0\}=3$$

$$\mathcal{I}_3(I)=\frac{2+3+0}{3\cdot 4}=\frac{5}{12} \qquad \square$$

As expected, the measures of incoherence just defined are monotonic w.r.t. the ordering between L-interpretations.

Proposition 8: Let I and J be two interpretations, if $I\leq J$, then $\mathcal{I}_i(I)\leq\mathcal{I}_i(J)$ for $i=1,2,3$.

Example 9: (Ex. 8 continued) Consider the interpretation

x	$J(x)$	$J(\sim x)$
p	$\bot$	c
q	b	b
r	a	$\bot$

Notice that $J \leq I$, therefore necessarily the measures of incoherence for J have to be lesser or equal than for I. Effectively, the measures of incoherence for J are:

$$\mathcal{I}_1(J) = \frac{1}{3} \quad ; \quad \mathcal{I}_2(J) = \sup\{0, 2, 0\} = 2$$

$$\mathcal{I}_3(J) = \frac{0+2+0}{3 \cdot 4} = \frac{1}{6} \qquad \square$$

B. Measuring Incoherence in Stable General Residuated Logic Programs

In this section, we use the measures defined in the previous section in order to determine how incoherent is a stable residuated logic program. Hence, we extend the measures $\mathcal{I}_1, \mathcal{I}_2$ and $\mathcal{I}_3$ to be applied to stable programs as follows:

$$\mathcal{I}_i(\mathbb{P}) = \inf\left\{\mathcal{I}_i(I) \colon I \text{ is a stable model of } \mathbb{P}\right\} \qquad (4)$$

It is straightforward to note that if $\mathbb{P}$ is consistent, then $\mathcal{I}_i(\mathbb{P}) = 0$ for $i = 1, 2, 3$.

Example 10: Recall the general residuated logic program described in Ex. 2:

$$\langle p \leftarrow \neg \sim p \quad ; \vartheta \rangle$$

$$\langle \sim p \leftarrow \neg p \quad ; \vartheta \rangle$$

which is stable independently from the value of ϑ.

Let us consider the case $\vartheta = 0.6$. We saw that $\mathbb{P}$ is inconsistent since none of its stable models

$$M_\delta \equiv \{(p, \delta); (\sim p, 1-\delta)\} \quad \text{for} \quad 0.4 \leq \delta \leq 0.6$$

is coherent. Firstly, we must consider the coherent pairs of p w.r.t. $\sim$ and each M_δ:

$$\Delta^{\sim}_{M_\delta}(p) = \left\{(x, y) \ : x \leq \delta, \ y \leq 1-\delta, \ y \leq \frac{3}{5} - \frac{1}{2}x\right\}$$

If we consider the Euclidean information measure in $[0, 1]$, each $\mathcal{I}(p; M_\delta)$ coincides with the infimum of the function:

$$f(x, y) = I(p) - x + I(\sim p) - y = \delta - x + 1 - \delta - y = 1 - x - y$$

in $\Delta^{\sim}_{M_\delta}(p)$. The reader can check easily that:

$$\mathcal{I}(p; M_\delta) = \frac{2}{5} - \frac{\delta}{2}$$

Once the measure $\mathcal{I}(p; M_\delta)$ has been obtained for each stable model M_δ, we can calculate the measures of incoherence for $\mathbb{P}$ as follows:

$$\mathcal{I}_1(\mathbb{P}) = \frac{1}{1} = 1$$

$$\mathcal{I}_2(\mathbb{P}) = \inf\left\{\frac{2}{5} - \frac{\delta}{2} \colon 0.4 \leq \delta \leq 0.6\right\} = 0.1$$

$$\mathcal{I}_3(\mathbb{P}) = \mathcal{I}_2(\mathbb{P}) = 0.1$$

From these measures we can deduce that every propositional symbol in $\mathbb{P}$ is incoherent ($\mathcal{I}_1(\mathbb{P}) = 1$) but *not too much* (since $\mathcal{I}_2(\mathbb{P}) = \mathcal{I}_3(\mathbb{P}) = 0.1$). $\square$

The measures defined above allow us to establish how incoherent a program is, but they say nothing about the reason for incoherence. The underlying idea in incoherence is contradictory information, and this information is generated by the rules. Therefore, the reason for incoherence is within the rules. As a result, we will define incoherence measures for sets of rules with the aim of determining what rule/s can be removed in order to obtain a coherent program.

Given an inconsistent stable program $\mathbb{P}$, we define the following measures of incoherence relative to a set of rules $X \subseteq \mathbb{P}$

$$\mathcal{I}_i(X; \mathbb{P}) = \begin{cases} 1 - \frac{\mathcal{I}_i(\mathbb{P} \smallsetminus X)}{\mathcal{I}_i(\mathbb{P})} & \text{if } \mathbb{P} \smallsetminus X \textit{ is stable} \\ -\infty & \text{otherwise} \end{cases}$$

Note, that the measure $\mathcal{I}_i(X; \mathbb{P})$ represents the amount of incoherence caused by the set of rules X in $\mathbb{P}$. The values of $\mathcal{I}_i(X; \mathbb{P})$ belong to $[-\infty, 1]$ and are interpreted as follows:

- If $\mathcal{I}_i(X; \mathbb{P}) = 1$, then rules in X are totally incoherent with the rest of the program. Therefore, we can restore the coherence of $\mathbb{P}$ by deleting all the rules in X.
- If $\mathcal{I}_i(X; \mathbb{P}) \in (0, 1)$, then rules in X are partially incoherent with the rest of the program. Hence, we can reduce the incoherence by deleting all the rules in X.
- If $\mathcal{I}_i(X; \mathbb{P}) = 0$, then rules in X are not the reason of the incoherence in $\mathbb{P}$.
- If $\mathcal{I}_i(X; \mathbb{P}) < 0$, then the subset X of rules is relevant to both coherence and/or stability of the program.

Note finally that this measure does not apply to consistent programs, since in this case the quotient $\mathcal{I}_i(\mathbb{P} \smallsetminus X)/\mathcal{I}_i(\mathbb{P})$ does not make sense.

Example 11: Consider the following stable program

$$r_1 \colon \langle p \leftarrow \neg(\sim p) \quad ; 1.0 \rangle \quad r_2 \colon \langle \sim p \leftarrow \neg p * \neg q \quad ; 1.0 \rangle$$
$$r_3 \colon \langle q \leftarrow \quad ; 0.8 \rangle \quad r_4 \colon \langle \sim q \leftarrow \quad ; 0.6 \rangle$$
$$r_5 \colon \langle \sim q \leftarrow p \quad ; 0.4 \rangle \quad r_6 \colon \langle \sim q \leftarrow \neg p \quad ; 1.0 \rangle$$

on $([0, 1], \leq, *_P, \leftarrow_P, \neg, \sim)$, where $\dot{\neg} = n_0$ and $\dot{\sim} = n_{0.5}$.

The program has a unique stable model, namely:

$$M \equiv \{(p, 1); (\sim p, 0); (q, 0.8); (\sim q, 0.6)\}$$

which is incoherent since $M(\sim q) = 0.6 \not\leq \dot{\sim}(M(q)) = 0$. If we consider the Euclidean information measure then :

$$\mathcal{I}_1(\mathbb{P}) = 0.5 \ ; \ \mathcal{I}_2(\mathbb{P}) = 0.4 \ ; \ \mathcal{I}_3(\mathbb{P}) = 0.2$$

The table below shows the measures of incoherence for some sets of rules:

X	$\mathcal{I}_1(X; \mathbb{P})$	$\mathcal{I}_2(X; \mathbb{P})$	$\mathcal{I}_3(X; \mathbb{P})$
$\{r_2\}$	0	0	0
$\{r_1, r_2\}$	0	-1	-1
$\{r_3\}$	$-\infty$	$-\infty$	$-\infty$
$\{r_4\}$	0	0.5	0.5
$\{r_1, r_3\}$	1	1	1
$\{r_2, r_3\}$	1	1	1

From the values above we can conclude that r_2 by itself does not cause the incoherence measured by $\mathcal{I}_1(\mathbb{P}), \mathcal{I}_2(\mathbb{P})$ and $\mathcal{I}_3(\mathbb{P})$; however, deleting both r_1 and r_2 increases the incoherence; rule r_3 is needed to guarantee stability; if we delete the rule $\{r_4\}$ from $\mathbb{P}$ we do not recover coherence but incoherence is reduced; finally, we recover coherence by deleting either the sets of rules $\{r_1, r_3\}$ or $\{r_2, r_3\}$. $\square$

The measures of incoherence defined in this section have adequate properties when applied on extended residuated logic programs: firstly, because the computation is highly simplified since its unique stable model is the least one; secondly, because that measures are monotonic w.r.t. the order between extended programs.

Definition 15: Let $\mathbb{P}, \mathbb{Q}$ be two extended residuated logic programs. We say that $\mathbb{P}$ is smaller or equal than $\mathbb{Q}$, denoted $\mathbb{P} \leq \mathbb{Q}$, if and only if for each rule $\langle p \leftarrow \mathcal{B}; \vartheta \rangle$ in $\mathbb{P}$ there is a rule $\langle p \leftarrow \mathcal{B}; \overline{\vartheta} \rangle$ in $\mathbb{Q}$ such that $\vartheta \leq \overline{\vartheta}$.

Proposition 9: Assume $\mathbb{P} \leq \mathbb{Q}$, then $\mathcal{I}_i(\mathbb{P}) \leq \mathcal{I}_i(\mathbb{Q})$ for all $i = 1, 2, 3$.

Corollary 1: If $\mathbb{P}$ is an extended residuated logic program, then for every set of rules X of $\mathbb{P}$

$$\mathcal{I}_i(X; \mathbb{P}) \in [0, 1] \quad \textit{for } i = 1, 2, 3$$

As a result, removing any set of rules from an incoherent extended program never increases the measure of incoherence. By the monotonicity stated by the proposition above implies that coherence can be only restored by removing information.

VI. Measuring Instability of Unstable Residuated Logic Programs

In this section we define measures of the amount of information which causes instability. In this way, as we described in section IV, we have two different possibilities to make that: to measure the excess of information and/or the lack of information in the logic program. Therefore, we divide the section into two parts:

- We start by defining a measure determining the minimum amount of information which it is necessary *to remove* from an unstable program so that it becomes stable (measuring the excess of information).
- Then, we define a measure which determines the minimum amount of information which is necessary *to add* to an unstable program so that it becomes stable (measuring the lack of information).

A. Measuring Instability of General Residuated Programs by Removing Information

Contrariwise to the classical case, in which the only way to delete information from a program is by deleting rules completely, in our framework we can just reduce their weights by some amount.

For that purpose, we need to fix a t-norm t to handle the values of L (recall that a t-norm is a commutative, associative, and monotonic map $L \times L \to L$ satisfying $t(1, x) = x$). Fixed such a t-norm, we define an operator to modify the weights of rules.

Consider a general residuated logic program $\mathbb{P}$, a set $X = \{\langle r_i; \vartheta_i \rangle\}_i$ of rules in $\mathbb{P}$, and a set of values $\{\varphi_i\}_i$ in L ; a new general residuated logic program is defined as

$$\Theta_{\mathbb{P}}(X, \{\varphi\}_i) = (\mathbb{P} \setminus X) \cup \{\langle r_i; t(\vartheta_i, \varphi_i) \rangle\}_i$$

In other words, the operator $\Theta_{\mathbb{P}}$ substitutes the weight of any rule $\langle r_j; \vartheta_j \rangle$ in X by $t(\vartheta_j, \varphi_j)$.

It is not difficult to note that the resulting program has smaller weights than the original one. The following example illustrates $\Theta_{\mathbb{P}}$ at work:

Example 12: Consider the residuated lattice with negation $([0, 1], \leq, *, \leftarrow, n)$, and the following residuated program

$$r_1 \colon \langle p \leftarrow q * t * \neg u \ ; \quad 0.7 \rangle \qquad r_2 \colon \langle p \leftarrow t * \neg s \ ; \quad 0.8 \rangle$$
$$r_3 \colon \langle q \leftarrow \neg v \ ; \quad 0.2 \rangle \qquad r_4 \colon \langle t \leftarrow s * u * \neg v \ ; \quad 0.9 \rangle$$

Assume the product t-norm ($t(x, y) = x \cdot y$) as the t-norm associated to the operator $\Theta_{\mathbb{P}}$. Then, the program $\Theta_{\mathbb{P}}(\{r_1, r_4\}, \{0.5, 0.9\})$ is shown below:

$$r_1 \colon \langle p \leftarrow q * t * \neg u \ ; \quad 0.35 \rangle \qquad r_2 \colon \langle p \leftarrow t * \neg s \ ; \quad 0.8 \rangle$$
$$r_3 \colon \langle q \leftarrow \neg v \ ; \quad 0.2 \rangle \qquad r_4 \colon \langle t \leftarrow s * u * \neg v \ ; \quad 0.81 \rangle$$

Notice that the weights of rules r_1 and r_4 are reduced by a factor 0.5 and 0.9 respectively. □

Using an arbitrary t-norm different from $*$ allows us greater liberty to reduce the weights. For example, in the residuated lattice $([0, 1], \leq, *, \leftarrow)$ where $*$ is defined by:

$$x * y = \begin{cases} x & \text{if } y = 1 \\ y & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases}$$

we can use the product t-norm in order to reduce gradually the weights in $\mathbb{P}$. Note that if we used $*$ to reduce the weights, then the reduction would be excessively drastic.

The measure of instability will be defined in terms of the amount of discarded information needed to get stability, and this will be computed by means of an information measure, see Section IV, and the formula

$$\sum_{i \in \mathbb{I}} \big(m(1) - m(\varphi_i)\big)$$

The sum above, in some sense, measures the amount of information discarded from the program; the lesser the values of φ_i, the more information discarded and greater the sum. Notice as well that $\Theta_{\mathbb{P}}$ does not reduce the weights of the program if φ_i is 1 for all i, and then the sum equals 0.

Example 13: Continuing with Example 12, if we consider in $[0, 1]$ the Euclidean information measure, then the amount of discarded information by the use of $\Theta_{\mathbb{P}}(\{r_1, r_4\}, \{0.5, 0.9\})$ would be $(1 - 0.5) + (1 - 0.9) = 0.6$. □

Now, given a general residuated logic program $\mathbb{P}$ and a set of rules $X = \{\langle r_i, \vartheta_i \rangle\}_i \subseteq \mathbb{P}$, we can define the following measure of instability as:

$$\textsc{Inst}^{-}_{\mathbb{P}}(X) = \inf_{\{\varphi_i\}_i} \left\{ \sum_{i \in \mathbb{I}} (m(1) - m(\varphi_i)) \mid \Theta_{\mathbb{P}}(X, \{\varphi_i\}_i) \textit{ is stable} \right\}$$

It is important to note that this operator might not be defined for some set of rules (in the case that $\Theta_{\mathbb{P}}(X, \{\varphi_i\}_i)$ is unstable for any choice of $\{\varphi_i\}_i$). This is not a big problem, as that would indicate that it is not possible to recover stability even by discarding completely all the rules in X.

Example 14: Let us consider the following logic program:

$$\begin{array}{ll} r_1: & \langle p \leftarrow s * \neg q \quad ; 0.8\rangle \\ r_2: & \langle q \leftarrow \neg r * \neg u \quad ; 0.8\rangle \\ r_3: & \langle r \leftarrow \neg p \quad ; 0.5\rangle \\ r_4: & \langle s \leftarrow \quad ; 0.8\rangle \\ r_5: & \langle t \leftarrow \neg p * \neg s \quad ; 0.5\rangle \\ r_6: & \langle v \leftarrow u * \neg r \quad ; 0.7\rangle \end{array}$$

defined on $([0,1], \leq, *_P, \leftarrow_P, n_{0.4})$.

In order to check that this program does not have stable models we will proceed by *reductio ad absurdum*. Let M be a stable model of $\mathbb{P}$. As s appears in the head of only one rule (actually a fact), then $M(s) = 0.8$. Moreover, as u does not occur in the head of any rule, $M(u) = 0$. We distinguish two cases now, we assume firstly that $M(p) \leq 0.4$. Then, as r appears in the head of only one rule, $M(r) = n_{0.4}(M(p)) \cdot 0.5 = 0.5$. Similarly for q and p we obtain $M(q) = n_{0.4}(M(r)) \cdot n_{0.4}(M(u)) \cdot 0.8 = 0$ and $M(p) = M(s) \cdot n_{0.4}(M(q)) \cdot 0.8 = 0.64 > 0.4$. Let us assume now that $M(p) > 0.4$. Then by using the same reasoning above we obtain the equalities: $M(r) = n_{0.4}(M(p)) \cdot 0.5 = 0$, $M(q) = n_{0.4}(M(r)) \cdot n_{0.4}(M(u)) \cdot 0.8 = 0.8$ and $M(p) = M(s) \cdot n_{0.4}(M(q)) \cdot 0.8 = 0 \leq 0.4$.

We will use the product t-norm and the Euclidean norm in the formulas above to measure the unstability of any of the rules individually.

For rule r_1, one can check that if its weight would set to $\alpha \leq 0.5$, then the program would have at least one (actually unique) fuzzy answer set, namely,

$$M \equiv \{(p, 0.8 \cdot \alpha); (q, 0); (r, 0.5); (s, 0.8); (t, 0); (v, 0)\}$$

On the other hand, if the weight would be a value $\alpha > 0.5$, then the program would keep being unstable (by proceeding as above).

It is possible to set the weight of r_1 to 0.5 using the factor $\varphi = 0.625$. Therefore, the least amount of information to be discarded from r_1 has to be $1 - 0.625 = 0.375$. In other words, $\text{INST}^-_{\mathbb{P}}(\{r_1\}) = 0.375$. Similarly, we can obtain the instability measures for the rest of rules:

x	$\text{INST}^-_{\mathbb{P}}(\{x\})$
r_2	0.5
r_3	0.2
r_4	0.375

For rules r_5 and r_6 it is not possible to get a stable program by reducing its weights. Notice that these results state that, in recovering stability by modifying just one rule, we need to discard much more information from r_2 than from r_3. □

Some results about the instability measure $\text{INST}^-_{\mathbb{P}}$ will be presented in Section VI-C.

B. Measuring Instability of General Residuated Logic Programs by Adding Information

In a dual manner to the above measure of instability, in this section we define another measure based on the amount of information that has to be *added* to an unstable program so that it gets stable.

We start in this case by fixing a t-conorm s to handle the values of L (recall that a t-conorm is a commutative, associative and monotonic map $L \times L \to L$ satisfying $s(1, x) = 1$ and $s(0, x) = x$). Fixed such a t-conorm, we define the following operator, which modifies the weights of rules.

Consider a general residuated logic program $\mathbb{P}$, a set $X = \{\langle r_i; \vartheta_i\rangle\}_i$ of rules in $\mathbb{P}$, and a set of values $\{\varphi_i\}_i$ in L; a new normal residuated logic program is defined as follows:

$$\Omega_{\mathbb{P}}(X, \{\varphi_i\}_i) = (\mathbb{P} \setminus X) \cup \{\langle r_i; s(\vartheta_i, \varphi_i)\rangle\}_i$$

In other words, the operator $\Omega_{\mathbb{P}}$ changes the weights of any rule $\langle r_j; \vartheta_j\rangle$ in X by $s(\vartheta_j, \varphi_j)$.

Notice that the operator $\Omega_{\mathbb{P}}$ increases the weights of the rules in X. Moreover, the higher the values φ_i, the higher the new weights of the rules.

Example 15: Recall the program given in Example 12. Consider the t-conorm to be $s(x, y) = \min\{x + y, 1\}$. Then, the modified program $\Omega_{\mathbb{P}}(\{r_1, r_3\}, \{0.4, 0.7\})$ is:

$$\begin{array}{llll} r_1: \langle p \leftarrow q * t * \neg u & ; & 1\rangle & r_2: \langle p \leftarrow t * \neg s \quad ; \quad 0.8\rangle \\ r_3: \langle q \leftarrow \neg v & ; & 0.9\rangle & r_4: \langle t \leftarrow s * u * \neg v \quad ; \quad 0.9\rangle \end{array}$$

The application of the operator $\Omega_{\mathbb{P}}$ results in changing the weight of r_1 by $\min\{0.7 + 0.4, 1\} = 1$ and that of r_3 by $\min\{0.2 + 0.7, 1\} = 0.9$. □

Anyway, it is important to note that increasing the weight of rules in an unstable program might not be enough to recover stability. We already stated in Section IV that instability might be due to missing facts or rules: the following pathological example exhibits this behaviour.

Example 16: On $([0,1], \leq, *_G, \leftarrow_G, n_{0.5})$, the program consisting of just the rule $\langle p \leftarrow n_{0.5}(p) \; ; 0.7\rangle$ cannot be made stable by simply increasing the weight of its rule. □

For measuring the minimal amount of information needed to recover stability, we need to consider the possible inclusion of new facts, rules or new literals in the bodies of existing rules. However, considering all possible combinations is certainly impractical from a computational point of view. We will see below how this process can be greatly simplified, as it is only necessary for our purposes to take into account just the inclusion of new facts and, thus, the inclusion of new literals in the bodies or new rules can be completely neglected.

Let us assume that when introducing occurrences of a new literal ℓ, either positively or negatively, in the bodies of some rules $\{\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle\}_i \subseteq \mathbb{P}$ the resulting program is stable. That is, the program $\mathbb{P}^*$ obtained from $\mathbb{P}$ by substituting the rules $\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle$ by either $\langle \ell_i \leftarrow \mathcal{B}_i * \ell; \vartheta_i\rangle$ or $\langle \ell_i \leftarrow \mathcal{B}_i * \neg\ell; \vartheta_i\rangle$ is stable. In any case, if M were a model of $\mathbb{P}^*$, then M would be as well a model of the program obtained from $\mathbb{P}$ by substituting rules $\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle$ by either $\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i * M(\ell)\rangle$ or $\langle \ell_i \leftarrow \mathcal{B}_i *; \vartheta_i * \dot{\neg}(M(\ell))\rangle$ (depending on whether ℓ is introduced positively or negatively in the rule). Note that, in the latter case, we have really decreased the weights of the rules, independently from ℓ and, in this case we have just removed information from the program since $\vartheta_i \geq \vartheta_i * M(\ell)$ and $\vartheta_i \geq \vartheta_i * \dot{\neg}(M(\ell))$. This is the case already studied in section above, hence $\text{INST}^-_{\mathbb{P}}$ also represents a possible excess of information in $\mathbb{P}$.

Secondly, assume now that when introducing a number of new rules $\{\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle\}_i$ in $\mathbb{P}$ the resulting program turns out to be stable. Assume that M is a stable model of $\mathbb{P} \cup \{\langle \ell_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle\}_i$. Again, we could obtain a stable program by simply including a number of new facts in $\mathbb{P}$, namely $\{\langle \ell_i \leftarrow ; M(\mathcal{B}_i) * \vartheta_i\rangle\}_i$. Note that, in the latter case, we need the same number of facts than rules introduced and, moreover, the required weights for the facts are lesser than those of the rules. In conclusion, in the latter case we have to add less information to the program in order to recover stability.

As a result of the above paragraphs, since our aim is to measure the minimum amount of information required to recover stability, we just need to take into account the possible addition of new facts not already included in the program.

Remark 6: It is convenient to recall that the semantics provided by $\mathbb{P} \cup \{\langle p_i \leftarrow ; M(\mathcal{B}_i) * \vartheta_i\rangle\}_i$ and $\mathbb{P} \cup \{\langle p_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle\}_i$ can be different. However, although this is an important feature that has to be considered, it is not within the scope of this work. The inclusion of new rules (or facts) has the single aim of stabilizing the program, and not detecting the *real* missing information.

For technical reasons, in order to define more easily the measure of required information to recover stability, given a program $\mathbb{P}$, we will consider its completion $\overline{\mathbb{P}}$, defined as

$$\overline{\mathbb{P}} = \mathbb{P} \cup \{\langle \ell_i \leftarrow ; 0\rangle \colon \ell_i \in \mathit{Lit}_{\mathbb{P}} \text{ and } \langle \ell_i \leftarrow ; \vartheta\rangle \notin \mathbb{P}\}$$

This results in explicitly including facts for all the symbols occurring in the program and, this way, we can add information to any of them by means of the operator $\Omega_{\overline{\mathbb{P}}}$ defined above. Obviously, the semantics of $\mathbb{P}$ and $\overline{\mathbb{P}}$ are equivalent.

The more information needed to recover stability, the more unstable the program is. The amount of information included when considering $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\}_i)$ is obtained by means of an information measure m and the formula

$$\sum_{i \in \mathbb{I}} m(\varphi_i)$$

Given a general residuated logic program $\mathbb{P}$, we define the instability measure of a set of rules $X = \{\langle r_i, \vartheta_i\rangle\}_i \subseteq \overline{\mathbb{P}}$ by:

$$\textsc{Inst}^+_{\mathbb{P}}(X) = \inf_{\{\varphi_i\}_i} \left\{ \sum_{i \in \mathbb{I}} m(\varphi_i) \colon \Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\}_i) \textit{ is stable} \right\}$$

Note that the definition above provides the minimum amount of information required to stabilize the program with regard to a given set of rules of the completion of the program.

It is important to note that the measure can be undefined for a given set of rules, and this would mean that stability cannot be reached by modifying just that set of rules. Example 16 shows this situation, since $\textsc{Inst}^+_{\mathbb{P}}(\{r_1\})$ is undefined.

C. Properties of the Instability Measures

In this section we provide some results concerning measures $\textsc{Inst}^-_{\mathbb{P}}$ and $\textsc{Inst}^+_{\mathbb{P}}$. The first one establishes antitonicity for the measures of instability.

Proposition 10: Let $\mathbb{P}$ be a general residuated logic program and let $X \subseteq Y$ be two sets of rules of $\mathbb{P}$ (resp. of $\overline{\mathbb{P}}$), then $\textsc{Inst}^-_{\mathbb{P}}(X) \geq \textsc{Inst}^-_{\mathbb{P}}(Y)$ (resp. $\textsc{Inst}^+_{\mathbb{P}}(X) \geq \textsc{Inst}^+_{\mathbb{P}}(Y)$).

The proposition above might look counter-intuitive at a first sight, however, this is not so. The reason is that the same amount of information removed (resp. added) from X would make the program stable when considered as a modification of the rules in Y; but it might be the case that less information could be removed (resp. added) from Y, perhaps distributed among the new rules in Y not in X, making the program stable.

Hence, we can obtain lower bounds for the measures $\textsc{Inst}^-_{\mathbb{P}}$ and $\textsc{Inst}^+_{\mathbb{P}}$ as follows:

Corollary 2: $\textsc{Inst}^-_{\mathbb{P}}(\mathbb{P})$ and $\textsc{Inst}^+_{\mathbb{P}}(\overline{\mathbb{P}})$ are defined for any general residuated logic program $\mathbb{P}$. Moreover, if $\textsc{Inst}^-_{\mathbb{P}}(X)$ and $\textsc{Inst}^+_{\mathbb{P}}(X)$ are defined, then

$$\textsc{Inst}^-_{\mathbb{P}}(X) \geq \textsc{Inst}^-_{\mathbb{P}}(\mathbb{P}) \quad \text{and} \quad \textsc{Inst}^+_{\mathbb{P}}(X) \geq \textsc{Inst}^+_{\mathbb{P}}(\overline{\mathbb{P}}).$$

The following proposition provides a useful theoretical result which implies an interesting corollary. It states that, although stability is not equivalent to null instability measure, the former implies the latter and, whenever the program has null instability measure it is possible to recover stability by removing or by adding an amount of information below any prescribed bound.

Proposition 11: Let $\mathbb{P}$ be a general residuated logic program and X a set of rules of $\mathbb{P}$:

- If $\mathbb{P}$ is stable then $\textsc{Inst}^-_{\mathbb{P}}(X) = \textsc{Inst}^+_{\mathbb{P}}(X) = 0$ for all X.
- If $\textsc{Inst}^-_{\mathbb{P}}(X) = 0$, then for all $\varepsilon > 0$ there exists a set $\{\varphi_i\}_i \subseteq L$ such that $\Theta_{\mathbb{P}}(X, \{\varphi_i\}_i)$ is stable and $\sum_{i \in \mathbb{I}} \big(m(1) - m(\varphi_i)\big) < \varepsilon$.
- If $\textsc{Inst}^+_{\mathbb{P}}(X) = 0$, then for all $\varepsilon > 0$ there exists a set $\{\varphi_i\}_i$ of values in L such that $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\}_i)$ is stable and $\sum_{i \in \mathbb{I}} m(\varphi_i) < \varepsilon$.

An interesting case occurs when the underlying lattice of truth-values is finite, since in this case stability coincides with null instability measure.

Corollary 3: Let $\mathbb{P}$ be a general logic program defined over a *finite* residuated lattice, and X a set of rules of $\mathbb{P}$. The following statements hold:

- There exists a set $\{\varphi_i\}_i$ of values in L such that $\textsc{Inst}^-_{\mathbb{P}}(X) = \sum_i (m(1) - m(\varphi_i))$ and $\Theta_{\mathbb{P}}(X, \{\varphi_i\}_i)$ is stable.
- There exists a set $\{\varphi_i\}_i$ of values in L such that $\textsc{Inst}^+_{\mathbb{P}}(X) = \sum_i m(\varphi_i))$ and $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\}_i)$ is stable.
- If there is a set of rules Y such that $\textsc{Inst}^-_{\mathbb{P}}(Y) = 0$, then $\mathbb{P}$ is stable.
- If there is a set of rules Y such that $\textsc{Inst}^+_{\mathbb{P}}(Y) = 0$, then $\mathbb{P}$ is stable.
- $\mathbb{P}$ is stable iff $\textsc{Inst}^-_{\mathbb{P}}(\mathbb{P}) = 0$ iff $\textsc{Inst}^+_{\mathbb{P}}(\mathbb{P}) = 0$.

D. Characterizing Instability in Terms of Stable Models

In this section we show that computing the values $\textsc{Inst}^-_{\mathbb{P}}(X)$ and $\textsc{Inst}^+_{\mathbb{P}}(X)$ is equivalent to obtain the set of stable models of two specific general residuated logic programs.

Let $\mathbb{P}$ be a general residuated logic program defined over the residuated lattice with negations $(L, \leq, *, \leftarrow, \neg, \sim)$ and let $X = \{\langle r_i; \vartheta_i\rangle\}$ be a set or rules in $\mathbb{P}$. We will define

two different general residuated logic programs $\mathbb{P}_X^-$ and $\mathbb{P}_X^+$ defined over a more general residuated lattice. We will show that the stable models of these two programs will allow us to determine the measures of $\text{INST}_{\mathbb{P}}^-(X)$ and $\text{INST}_{\mathbb{P}}^+(X)$. The procedure is introduced in detail as follows:

The computation of instability measures for a set of rules X depends on the existence of values $\varphi_i \in L$ satisfying that $\Theta_{\mathbb{P}}(X, \{\varphi_i\})$ (resp. $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\})$) is stable.

The introduction of the values φ_i into the play is done by considering, for each rule $\langle r_i; \vartheta_i\rangle \in X$, two *fresh* propositional symbols α_i and β_i and the pair of rules

$$\langle \alpha_i \leftarrow \neg_{\mathcal{N}}(\beta_i) \quad ; 1\rangle \tag{5}$$

$$\langle \beta_i \leftarrow \neg_{\mathcal{N}}(\alpha_i) \quad ; 1\rangle \tag{6}$$

where the negation associated with $\neg_{\mathcal{N}}$ is an *involutive* negation $\mathcal{N}$ defined on L.

It is known that not every residuated lattice admits an involutive negation, but it is possible to embed L into another lattice $\overline{L}$ on which $\mathcal{N}$ exists [8]. In $\overline{L}$, we have that $x \leq y$ for all $x \in \overline{L} \setminus L$ and $y \in L$ and, moreover, every adjoint pair and negation operator can be extended to $\overline{L}$.

As a result of the previous considerations, the residuated logic program $\mathbb{P}$ can be extended to $\overline{L}$ in a way such that every stable model of $\mathbb{P}$ (interpreted on $(L, \leq, *, \leftarrow, \neg)$) is in fact a stable model in the extended lattice $\overline{L}$.

Let us continue with the rules introduced in (5), (6) above. The set of stable models of this pair of rules is given parametrically by the set $\{M_\lambda \equiv (\alpha_i, \lambda); (\beta, \mathcal{N}(\lambda)\}_{\lambda \in \overline{L}}$.

We are now in conditions to define the auxiliary programs $\mathbb{P}_X^-$ and $\mathbb{P}_X^+$ which will be used to compute the instability measures w.r.t. X; in both cases we consider the residuated implication associated to a given t-norm which, therefore, is assumed to be left-continuous..

1) $\mathbb{P}_X^-$ is built by considering:
   a) Every rule in $\mathbb{P}$ which does not belong to X.
   b) Substituting every rule $\langle \ell_i \leftarrow \mathcal{B}_i \, ; \vartheta_i\rangle \in X$ by rules (5), (6), (7) and (8)

$$\langle \ell_i \leftarrow \mathcal{B}_i * \gamma_i \quad ; 1\rangle \tag{7}$$

$$\langle \gamma_i \leftarrow_t \alpha_i \quad ; \vartheta_i\rangle \tag{8}$$

   where γ_i is a *fresh* propositional symbol, and $\leftarrow_t$ is the residuated implication associated to the t-norm t used in the measure $\text{INST}_{\mathbb{P}}^-$.

2) $\mathbb{P}_X^+$ is built by considering:
   a) Every rule in $\overline{\mathbb{P}}$ which does not belong to X
   b) Substituting each rule $\langle \ell_i \leftarrow \mathcal{B}_i \, ; \vartheta_i\rangle \in X$ by the rules (5), (6) and the following pair of rules:

$$\langle \ell_i \leftarrow \mathcal{B}_i * \neg_{\mathcal{N}}(\gamma_i) \quad ; 1\rangle \tag{9}$$

$$\langle \gamma_i \leftarrow_{t_s} \neg_{\mathcal{N}}(\alpha_i) \quad ; \mathcal{N}(\vartheta_i)\rangle \tag{10}$$

   where γ_i is a *fresh* propositional symbol, and $\leftarrow_{t_s}$ is the residuated implication associated to the t-norm $t_s(x, y) = \mathcal{N}(s(\mathcal{N}(x), \mathcal{N}(y)))$; where s is t-conorm used in the measure $\text{INST}_{\mathbb{P}}^+$.

We will only provide results for $\text{INST}_{\mathbb{P}}^+(X)$, as those for $\text{INST}_{\mathbb{P}}^-(X)$ are similar.

Lemma 2: Consider a general residuated logic program $\mathbb{P}$, let X be a set of rules of $\overline{\mathbb{P}}$, and M an $\overline{L}$-interpretation $M\colon Lit_{\mathbb{P}} \cup \{\alpha_i, \beta_i, \gamma_i\}_i \to \overline{L}$ satisfying $M(\beta_i) = \mathcal{N}(M(\alpha_i))$ and $M(\gamma_i) = t_s(\mathcal{N}(\vartheta_i), \mathcal{N}(M(\alpha_i)))$, then the following statements hold:

1) If N is a model of $(\mathbb{P}_X^+)_M$, then it is also a model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$.
2) Reciprocally, any model N of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$ can be extended to a model of $(\mathbb{P}_X^+)_M$.

Proposition 12: Let $\mathbb{P}$ be a general residuated logic program and let X be a set of rules of $\overline{\mathbb{P}}$.

1) If M is a stable model of $\mathbb{P}_X^+$, then $M_{|_{Lit_{\mathbb{P}}}}$ is also a stable model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})$.
2) Reciprocally, any stable model M of $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\})$ can be extended to a stable model of $\mathbb{P}_X^+$.

It is not difficult to check that Lemma 2 and Proposition 12 above can be restated by changing $\mathbb{P}_X^+$ by $\mathbb{P}_X^-$, and $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\})$ by $\Theta_{\mathbb{P}}(X, \{\varphi_i\})$.

Proposition 12 shows that there is a one-to-one correspondence among the stable models of $\mathbb{P}_X^+$ and the parameters φ_i such that $\Omega_{\overline{\mathbb{P}}}(\{X, \{\varphi_i\})$ is stable. As a result, we can compute $\text{INST}_{\mathbb{P}}^+(X)$ by means of the stable models of $\mathbb{P}_X^+$.

Corollary 4: Let $\mathbb{P}$ be a general residuated logic program, and consider that $SM(\cdot)$ represents the set of stable models of the corresponding program, then:

$$\text{INST}_{\mathbb{P}}^+(X) = \inf_{\substack{M \in SM(\mathbb{P}_X^+) \\ M(\alpha_i) \in L}} \left\{ \sum_i m(M(\alpha_i)) \right\}$$

$$\text{INST}_{\mathbb{P}}^-(X) = \inf_{\substack{M \in SM(\mathbb{P}_X^-) \\ M(\alpha_i) \in L}} \left\{ \sum_i (m(1) - m(M(\alpha_i))) \right\}$$

It important to note that programs $\mathbb{P}_X^-$ and $\mathbb{P}_X^+$ might not be stable. This would mean that the measures $\text{INST}_{\mathbb{P}}^-(X)$ and $\text{INST}_{\mathbb{P}}^+(X)$ are not defined for the set of rules X.

The following example shows how to apply the procedure above to a given program.

Example 17: Consider $([0, 1], \leq, *_P, \leftarrow_P, \neg)$, where the negation operator is defined as:

$$\neg(x) = \begin{cases} 1 & \text{if} \quad x \leq 0.5 \\ 1 - x & \text{if} \quad x > 0.5 \end{cases}$$

and the program below:

$$\begin{aligned} r_1\colon \langle p \leftarrow s * \neg q \quad ; 1\rangle \\ r_2\colon \langle q \leftarrow \neg r \quad ; 0.9\rangle \\ r_3\colon \langle r \leftarrow \neg p \quad ; 0.9\rangle \\ r_4\colon \langle s \leftarrow \neg t * \neg u \quad ; 1\rangle \end{aligned}$$

Let us see that $\mathbb{P}$ is unstable. We proceed by reductio ad absurdum by assuming that there is a fuzzy stable model M. As the propositional symbols t and u do not appear in the head of any rule, necessarily $M(t) = M(u) = 0$. As M is a stable model, M is the least model of the reduct $\mathbb{P}_M$. The reduct of r_4 coincides with the fact $\langle s \leftarrow ; 1 * \neg M(t) * \neg M(u)\rangle = \langle s \leftarrow ; 1\rangle$. So $M(s) = 1$. We distinguish now two cases. Firstly we assume that $M(q) > 0.5$. Then as p appears only in the head

of one rule, $M(p)$ is equal to $M(s)*\neg M(q)*1 = 1-M(q) < 0.5$. Similarly, we can infer that $M(r) = \neg M(p) * 0.9 = 0.9$ and then $M(q) = \neg M(r) * 0.9 = 0.1 * 0.9 < 0.5$, contradictory with the hypothesis. Assume now that $M(q) \leq 0.5$. Then by using a similar reasoning than in the previous case, $M(p) = M(s)*\neg M(q)*1 = 1$, thus $M(r) = \neg M(p)*0.9 = 0$ and then $M(q) = \neg M(r) * 0.9 = 1 * 0.9 > 0.5$, contradictory with the hypothesis. The instability will be measured by using $\text{INST}^+_{\mathbb{P}}$.

Let us fix the information measure and a t-conorm, say $m(x) = x^2$ and $s(x,y) = \max\{x,y\}$. The completion $\overline{\mathbb{P}}$ is obtained by adding the following rules

$$r_5\colon \langle p \leftarrow \ ; 0\rangle \qquad r_6\colon \langle q \leftarrow \ ; 0\rangle$$
$$r_7\colon \langle r \leftarrow \ ; 0\rangle \qquad r_8\colon \langle s \leftarrow \ ; 0\rangle$$
$$r_9\colon \langle t \leftarrow \ ; 0\rangle \qquad r_{10}\colon \langle u \leftarrow \ ; 0\rangle$$

For the construction of $\mathbb{P}^+_X$, we will consider the involutive negation $n(x) = 1 - x$. Note that we can consider any involutive negation in $[0,1]$, since the final result does not depend on this choice.

The reader can easily verify that $\text{INST}^+_{\mathbb{P}}(\mathbb{P})$ is undefined (by using a reasoning similar to prove that $\mathbb{P}$ has not stable models). This means that the reason why the program is not stable can only be due to some missing facts; therefore, it makes sense to consider $\text{INST}^+_{\mathbb{P}}(\{r_i\})$ for $i = 5, \ldots, 10$.

For the sake of this example, we will only compute the value of $\text{INST}^+_{\mathbb{P}}(r_9)$ in terms of the stable models of $\mathbb{P}^+_{r_9}$. This auxiliary program is obtained by considering a new default negation symbol $\neg_n$ associated to the negation operator n and by substituting r_9 in $\mathbb{P}$ by the following four rules:

$$\langle \alpha \leftarrow \neg_n(\beta) \ ; 1\rangle$$
$$\langle \beta \leftarrow \neg_n(\alpha) \ ; 1\rangle$$
$$\langle t \leftarrow \neg_n(\gamma) \ ; 1\rangle$$
$$\langle \gamma \leftarrow \neg_n(\alpha) \ ; 1\rangle$$

The family of stable models of this new program is:

$$SM(\mathbb{P}^+_{r_9}) = \big\{\{(p,1-\lambda);(q,0);(r,0.9);(t,\lambda);(u,0); \\ ;(s,1-\lambda);(\alpha,\lambda)\} \mid \lambda > 0.5\big\}$$

We only check that the only stable models are those in $SM(\mathbb{P}^+_{r_9})$. Let M be a stable model satisfying $M(t) \leq 0.5$. Then, by using the same reasoning done to check that $\mathbb{P}$ has not stable model, we obtain a contradiction.

Therefore:

$$\text{INST}^+_{\mathbb{P}}(r_9) = \inf\{m(M(\alpha)) \mid M \in SM(\mathbb{P}^+_{r_9})\} = \\ = \inf\{\lambda^2 \mid \lambda > 0.5\} = 0.25$$

and this means that we have to increase the value of t (given by rule r_9) by at least 0.5, which corresponds to an information measure of 0.25, in order to recover stability. □

VII. Measuring Inconsistence: Some Applications

Once the measures of incoherence and instability have been presented, we can study how to measure inconsistency. We have two different options, either to measure instability and incoherence independently, or to measure inconsistency directly by using the ideas described in Section VI for instability. These ideas can be easily extended to measure the inconsistency in general residuated logic program by defining:

$$\text{INCON}^{rem}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i\}) = \inf\{\sum_{i\in\mathbb{I}} m(1) - m(\varphi_i)\colon \\ O_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i,\{\varphi_i\}_i) \textit{ is consistent}\}$$

and

$$\text{INCON}^{add}_{\mathbb{P}}(\{\langle r_i,\vartheta_i\rangle\}_i\}) = \\ = \inf\{\sum_{i\in\mathbb{I}} m(\varphi_i)\colon \Omega_{\overline{\mathbb{P}}}(\{\langle r_i,\vartheta_i\rangle\}_i,\{\varphi_i\}_i) \textit{ is consistent}\}$$

Notice that, as stated in the previous section, the latter case might not be defined. Furthermore, contrariwise to the case of instability, we cannot even ensure the existence of a set of rules on which the measure $\text{INCON}^{add}_{\mathbb{P}}$ can be defined.

Obviously, in order to measure independently instability and incoherence of an inconsistent program we need to restrict our attention to finite programs, so that we can use Proposition 3. In this context, we can measure the inconsistency of a set of rules X and a general residuated logic programs $\mathbb{P}$, as follows:

1) Measure instability of $\mathbb{P}$.
2) Consider the set of stable programs with the form $O_{\mathbb{P}}(X,\{\varphi_i\})$ (resp. $\Omega_{\overline{\mathbb{P}}}(X,\{\varphi_i\}_i)$) such that $\sum_{i\in\mathbb{I}} m(1) - m(\varphi_i) = \text{INST}^+_{\mathbb{P}}(X)$ (resp. $\sum_{i\in\mathbb{I}} m(\varphi_i) = \text{INST}^+_{\mathbb{P}}(X)$).
3) Measure incoherence of the set of stable programs described in the step above.

In the subsequent sections, we will focus on some applications of the approach stated above.

A. Conflict Mediation: Negotiations

Negotiations arise naturally in everyday life, from choosing the TV channel at home to determining the renewal of an employment contract with our employer/employee. Fuzzy logic programming has the expressive power needed to cover all contexts in which conflicts occur, it seems an appropriate environment to define automatic processes of negotiation.

Let us see how to apply our framework to the problem of negotiating the price of an item. Consider that A wants to sell a certain item to B, and let the propositional symbol p represent the statement "*the object has high value*." Agent A will be interested in a high value for $M(p)$ (that is, the item would have a high price), whereas agent B will be interested in the opposite. Thus we can define the program

$$\langle p \leftarrow \ ; \vartheta_1\rangle \qquad \langle \sim p \leftarrow \ ; \vartheta_2\rangle$$

where ϑ_1 and ϑ_2 are the assigned values by A and B respectively. In some sense, ϑ_1 represents the minimum price accepted by the seller and ϑ_2 the maximum price that B is willing to pay. If the program were consistent, then the value assigned by the seller to the object would be less than what B is willing to pay, and the transaction would take place without problems. However, if the program were to be inconsistent there would be a conflict, and the transaction would not occur. This is where our approach would be applicable. Since the

program is extended, the only way to recover consistence is by deleting information, i.e. either A or B (or both) should reduce their claim. Inconsistency measures indicate how far we are from an agreement about the price, a high value of inconsistency would imply that the negotiation towards an agreement could be costly, whereas a low value would imply the opposite; this would be very useful should we have to decide whether or not to spend resources to eliminate the conflict. At this point, we can take advantage from using the measure $\text{INCON}_{\mathbb{P}}^{rem}$, since its value is associated to a consistent program that can be identified with the solution to the conflict. Furthermore, this solution has the property that is obtained by reducing the change from the initial positions of of A and B.

The previous example of negotiation can be of the type win-lose or lose-lose, depending on whether both weights are modified or only one of them. However, there is another type called win-win negotiation in which all parties gain from the resolution of the conflict. The following example shows a case of this kind of negotiation. Suppose we have two negotiators A and B who wish to establish a purchase of raw materials for a factory and we are working in the residuated lattice $([0,1], *_p, \leftarrow_p, \neg, \sim)$ where both negations are represented by the usual negation operator $1-x$. Agent A proposes that if the purchase is not made to supplier q_1, then it has to be done to supplier q_2; as a consequence, supplier q_2 is chosen by default, which can be encoded by using the following rule:

$$\langle q_2 \leftarrow \neg q_1 \ ; 1 \rangle$$

On the other hand, B proposes not to buy all the supplies to q_2 because some stuff is needed urgently, and supplier q_2 takes too long to send the material. Therefore, B introduces the rule:

$$\langle \sim q \leftarrow \ ; 0.5 \rangle$$

The resulting program has just one stable model,

$$M = \{(p,0), (\sim p, 0), (q,1), (\sim q, 0.5)\}$$

which is clearly incoherent. For a win-win negotiation, we can apply the inconsistency measure given by $\text{INCON}_{\mathbb{P}}^{add}$ to the set $\{\langle p \leftarrow \ 0 \rangle\}$ (with respect to the product t-norm and the information measure induced by the Euclidean norm), obtaining the value 0.5. By using the characterization given in Section VI-D, this value is related to the stable model $M = \{(p, 0.5), (\sim p, 0), (q, 0.5), (\sim q, 0.5)\}$ for the modified program obtained from the initial by including the rule

$$\langle \sim p \leftarrow \ ; 0.5 \rangle$$

As a result, A and B reach to a consensus as to buy half the material to q_1 and the other half to q_2. Notice that the conflict has been solved without reducing the pretensions of any party.

Obviously, not every negotiation is that simple. In fact, negotiations usually depend on factors, such as emotional intelligence or body language, which make difficult the automatization. However, it might be possible to create an expert system able to advise the negotiating strategies, to provide alternatives to conflict, according to mutual positions, to identify or infer the interests of the party, etc. It is in this kind of problems where our framework could find practical applications.

B. Managing Information from Different Sources

When building a database using data from different sources, it is quite common to obtain an inconsistent result. In most cases, this happens because the information coming from each source corresponds to a different point of view.

Assume we have a number of experts to provide us certain information in the form of a residuated logic program $\mathbb{P}_i$, each of them consistent. Finally, we put all the information together and build program $\mathbb{P}$ as the disjoint union of all $\mathbb{P}_i$ (that is, if a rule r belongs to n program $\mathbb{P}_i$, then r appears n times in $\mathbb{P}$). If $\mathbb{P}$ is consistent, then there is nothing to do; otherwise, if $\mathbb{P}$ is inconsistent, we consider the maximal consistent sets $\mathbb{Q}_j$ formed as unions of $\mathbb{P}_i$. The added value of our results to this approach is that we can assign each program $\mathbb{Q}_j$ a value that represents how arguable the information contained in $\mathbb{Q}_j$ is with respect to the rest of the program. Specifically, the above value is computed by

$$\min\{\text{INCON}_{\mathbb{P}}^{rem}(\mathbb{Q}_j^c), \text{INCON}_{\mathbb{P}}^{add}(\mathbb{Q}_j^c)\}$$

where $\mathbb{Q}_j^c$ denotes the set of rules $\mathbb{P} \setminus \mathbb{Q}_j$. Notice that this value is always defined, since $O_{\mathbb{P}}(\mathbb{Q}_j^c, \{0\}) = \mathbb{Q}_j$ is consistent. Furthermore, this value represents the amount of information that we have to modify in $\mathbb{Q}_j^c$ in order to obtain a consistent program, so the more information we have to modify in $\mathbb{Q}_j^c$, the more information is available at $\mathbb{Q}_j^c$ which contradicts (and therefore refutes) the information in $\mathbb{Q}_j$. As a result, our measures of inconsistency would help us to know which opinions are "more consistent" and difficult to refute.

C. Detecting False or Erroneous Information

When a knowledgebase has a large number of rules is not uncommon to find that it is inconsistent, and this is generally due to some incorrect rules. Moreover, there are environments where it is common to find false information, such as databases that report the statements of witnesses of a crime. Therefore locating the misleading or false information in a database $\mathbb{P}$ is an issue. With the inconsistency measures that have been defined we can set priorities on certain sets where to find the source of inconsistency.

Depending on the type of inconsistency, we will proceed in two different ways, if the cause is incoherence, then we consider the minimal inconsistent sets $\mathbb{P}_j$ formed by rules in $\mathbb{P}$. In some sense, these sets determine clusters of inconsistency.

Applying the inconsistency measures we would obtain information about the degree of inconsistency between the rules in $\mathbb{P}_j$. Furthermore, computing the values $\mathcal{I}_i(\mathbb{P}_j; \mathbb{P})$ we know how much incoherence is generated by $\mathbb{P}_j$ in $\mathbb{P}$. If we accept that false/wrong information is the cause of all the inconsistency in the program, we should be interested in determining the sets $\mathbb{P}_j$ such that $\mathcal{I}_i(\mathbb{P}_j, \mathbb{P}) = 1$.

However, it could be the case that inconsistency is due to several $\mathbb{P}_j$, so it is very likely that there are no minimal sets such that $\mathcal{I}_i(\mathbb{P}_j, \mathbb{P}) = 1$. Anyway, it is advisable to start the search for wrong information on the subsets $\mathbb{P}_j$ such that its measure $\mathcal{I}_i(\mathbb{P}_j, \mathbb{P})$ is closer to 1.

Should the program be unstable, we would be interested in finding minimal sets of rules $\mathbb{P}_j$ for which measures $\text{INCON}_{\mathbb{P}}^{rem}(\mathbb{P}_j)$ and $\text{INCON}_{\mathbb{P}}^{add}(\mathbb{P}_j)$ are defined, and

such that $\text{INST}^+_{\mathbb{P}}(\mathbb{P}_j) = \text{INCON}^{rem}_{\mathbb{P}}(\mathbb{P}_j)$ and $\text{INST}^+_{\mathbb{P}}(\mathbb{P}_j) = \text{INCON}^{add}_{\mathbb{P}}(\mathbb{P}_j)$.

VIII. Related Work

The content of this section is two-fold: on the one hand, we focus on the different extension, to our knowledge, of the answer set semantics; on the other hand, we comment on some alternative approaches to the measurement of inconsistency in knowledge-bases.

In recent years several extensions of the answer set semantics to some non-classical logics have been developed. We relate below our generalization to some approaches based on *Probabilistic Logic* [22] [40] [34], *Annotated Logic* [35], *Antitonic Logic* [4], *Fuzzy Description Logic* [23] [24] and *Fuzzy Logic* [17], [18], [38], [39].

- Most of the approaches are conservative, in that answer sets (or stable models) are generalized by using adequate versions of the GL-reduct. Exceptions are [39] and [40], interestingly neither approach proves that its generalized answer sets are minimal models.
- The definition of interpretation in most approaches is a mapping which assigns an element in the set of truth-values to each propositional symbol. Exceptions are [34] and [4]: in the former, interpretations assign an interval of truth values to each propositional symbol whereas, in the latter, interpretations assign two values, representing the degree of truth and the degree of non-falsity, to each propositional symbol.
- Our personal contribution here is the notion of coherence, linked to the consideration of the strong negation. Although other approaches allow strong negation in the syntax, our definition of coherence is more flexible than that given, for instance, in [40] in which two opposite literals (p and $\sim p$) cannot have a positive value simultaneously, an excessive restriction in some cases.

On the other hand, there are a couple of approaches related with the measures of inconsistency in the classical framework (see [13]). These approaches can be classified as based on Shanon's information theory [21], on possibilistic logic [6], on paraconsistent logic [11] or on a mixture of paraconsistent logic and Shanon's information theory [42]. However, the measures provided in this paper are introduced in a completely different way (which is not possible in the classical framework) since in fuzzy theories we can reduce the weights instead of completely delete the information provided for a rule. In classical approaches there is a common procedure: measures are obtained by considering either maximal consistent sets of rules or minimal inconsistent sets of rules. Note that considering such sets implies a complete deletion of the information of some rules in each maximal consistent set of rules; our approach avoids this complete deletion of rules by modifying its weights.

Although measures of inconsistency can be closely related to fuzzy logic, to the best of our knowledge, there are no approaches measuring the inconsistency in this more general framework. Perhaps this is due to the fact that there is not a consensus on how to generalize inconsistency in a fuzzy framework. For instance, [12], [39] consider α-inconsistent interpretations as interpretations which assign a value greater or equal than α to the formula $p \wedge \neg p$ for certain propositional symbol p; [40] considers that an interpretation is inconsistent if it assigns a positive value to both p and $\neg p$; [6] defines α-inconsistent interpretations as interpretations which assign a value greater or equal than α to the symbol $\bot$ (falsum).

IX. Conclusions and Future Work

We have considered the concept of inconsistency in two different dimensions; namely, *instability* (no stable model exists) and *incoherence* (any stable model is contradictory). Moreover, we have defined several measures of incoherence and instability for general residuated logic programs. Once the measures of incoherence and instability have been presented in Sections V and VI, we have convenient tools to measure inconsistency.

It is possible to analyze separately instability and incoherence for an inconsistent program, although we need to restrict the measures to finite lattices, so that we can use Corollary 3. Under this assumption, we can measure the inconsistency for a set of rules X in a general residuated logic program $\mathbb{P}$, as follows:

1) Measure instability of X in $\mathbb{P}$.
2) Consider the set of stable programs with the form $\Theta_{\mathbb{P}}(X, \{\varphi_i\})$ (resp. $\Omega_{\overline{\mathbb{P}}}(X, \{\varphi_i\}_i)$) satisfying $\sum_{i\in\mathbb{I}}(m(1) - m(\varphi_i)) = \text{INST}^-_{\mathbb{P}}(X)$ (respectively $\sum_{i\in\mathbb{I}} m(\varphi_i) = \text{INST}^+_{\mathbb{P}}(X)$).
3) Measure incoherence of the stable programs in 2) above.

There are several interesting lines on which this research can be continued: firstly, from a technical point of view, the computation of the measures $\text{INST}^-_{\mathbb{P}}(X)$ and $\text{INST}^+_{\mathbb{P}}(X)$ in terms of the stable models of $\mathbb{P}^-_X$ and $\mathbb{P}^+_X$ is a computationally hard work. Nevertheless, we do not need to know all the stable models, but only those which assign to the propositional symbols α_i values minimizing the sums $\sum_i m(M(\alpha_i))$ and $\sum_i m(1) - m(M(\alpha_i))$ respectively. We believe that this can be attempted by suitably adapting the top-k approaches given in [36] to our fuzzy answer set semantics.

A thorough theoretical treatment of measures of inconsistency has still to be done. Moreover, the underlying information measure assumed in this paper has to be further studied, perhaps by developing some kind of a fuzzy information theory.

From a practical point of view, we expect to apply the measures defined in this paper to some real-world problems.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science project TIN09-14562-C05-01 and Junta de Andalucía project and FQM-5233.

We thank the anonymous referees, whose comments and suggestions have significantly improved the original version of this paper.

REFERENCES

[1] M. Balduccini. CR-prolog as a specification language for constraint satisfaction problems. In *Lecture Notes in Artificial Intelligence*, volume 5753, pages 402–408, 2009.

[2] N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logic*, pages 7–37. Reidel Publishing Company, Boston, 1977.

[3] L. Bertossi, A. Hunter, and T. Schaub, editors. *Inconsistency Tolerance*, volume 3300 of *Lecture Notes in Computer Science*. Springer, 2005.

[4] C. V. Damásio and L. M. Pereira. Antitonic logic programs. *Lect. Notes in Artificial Intelligence*, 2173:379–392, 2001.

[5] C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. *Lect. Notes in Artificial Intelligence*, 2143:748–759, 2001.

[6] D. Dubois, S. Konieczny, and H. Prade. Quasi-possibilistic logic and its measures of information and conflict. *Fundamenta Informaticae*, 57(2-4):101–125, 2003.

[7] M. Fitting. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science*, 278:25–51, 1999.

[8] N. Galatos and J. G. Raftery. Adding involution to residuated structures. *Studia Logica*, 77(2):181–207, 2004.

[9] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP-88*, pages 1070–1080, 1988.

[10] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[11] J. Grant and A. Hunter. Measuring inconsistency in knowledgebases. *Journal of Intelligent Information Systems*, 27(2):159–184, 2006.

[12] P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Kluwer Academic, 1998.

[13] A. Hunter. Approaches to measuring inconsistent information. *Lecture Notes in Computer Science*, 3300:189–234, 2005.

[14] A. Hunter and S. Konieczny. Approaches to measuring inconsistent information. In *Inconsistency Tolerance*, volume 3300 of *Lecture Notes in Computer Science*, pages 191–236, 2005.

[15] A. Hunter and S. Konieczny. Shapley inconsistency values. In *Principles of knowledge representation and reasoning*, pages 249–259, 2006.

[16] A. Hunter and S. Konieczny. Measuring inconsistency through minimal inconsistent sets. In *Proc of Principles of Knowledge Representation and Reasoning (KR'08)*, pages 358–366. AAAI Press, 2008.

[17] J. Janssen, S. Heymans, D. Vermeir, and M. De Cock. Compiling fuzzy answer set programs to fuzzy propositional theories. *Lecture Notes in Computer Science*, 5336:362 – 376, 2008.

[18] J. Janssen, S. Schockaert, D. Vermeir, and M. De Cock. General fuzzy answer set programs. *Lecture Notes in Computer Science*, 5571(352–359), 2009.

[19] K. Knight. Measuring inconsistency. *Journal of Philosophical Logic*, 31(1):77–98, 2002.

[20] S. Konieczny, J. Lang, and P. Marquis. Quantifying information and contradiction in propositional logic through epistemic actions. In *International Joint Conference on Artificial Inteligence*, pages 106–111, 2003.

[21] E. L. Lozinskii. Resolving contradictions: A plausible semantics for inconsistent systems. *Journal of Automated Reasoning*, 12(1):1–32, 1994.

[22] T. Lukasiewicz. Many-valued disjunctive logic programs with probabilistic semantics. *Lect. Notes in Computer Science*, pages 277–289, 1999.

[23] T. Lukasiewicz. Fuzzy description logic programs under the answer set semantics for the semantic web. *Fundamenta Informaticae*, 82(3):289–310, 2008.

[24] T. Lukasiewicz and U. Straccia. Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web. *Lect. Notes in Computer Science*, pages 289–298, 2007.

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

[26] N. Madrid and M. Ojeda-Aciego. On coherence and consistence in fuzzy answer set semantics for residuated logic programs. *Lect. Notes in Computer Science*, 5571:60–67, 2009.

[27] N. Madrid and M. Ojeda-Aciego. On the measure of incoherence in extended residuated logic programs. In *IEEE Intl Conf on Fuzzy Systems (FUZZ-IEEE'09)*, pages 598–603, 2009.

[28] N. Madrid and M. Ojeda-Aciego. Measuring instability in normal residuated logic programs: adding information. In *IEEE Intl Conf on Fuzzy Systems (FUZZ-IEEE'10)*, pages 2244–2250, 2010.

[29] N. Madrid and M. Ojeda-Aciego. Measuring instability in normal residuated logic programs: discarding information. *Communications in Computer and Information Science*, 80:128–137, 2010.

[30] N. Madrid and M. Ojeda-Aciego. On the notion of coherence in fuzzy answer set semantics. In *XV Spanish Conference on Fuzzy Logic and Technology*, pages 157–162, 2010.

[31] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. *Lect. Notes in Artificial Intelligence*, 2173:351–364, 2001.

[32] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146:43–62, 2004.

[33] G. Priest. *Paraconsistent Logic*, volume 6 of *Handbook of philosophical logic*, pages 287–395. Springer, 2002.

[34] E. Saad. Towards the computation of stable probabilistic model semantics. *Lecture Notes in Computer Science*, 4314:143–158, 2006.

[35] U. Straccia. Annotated answer set programming. Technical Report 2005-TR-51, Istituto di Scienza e Tecnologie dell'Informazione-Consiglio Nazionale delle Ricerche, 2005.

[36] U. Straccia. Towards vague query answering in logic programming for logic-based information retrieval. *Lecture Notes in Computer Science*, 4529:125–134, 2007.

[37] U. Straccia. Managing uncertainty and vagueness in description logics, logic programs and description logic programs. *Lecture Notes in Computer Science*, 5224:54–103, 2008.

[38] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir. Computing fuzzy answer sets using dlvhex. *Lecture Notes in Computer Science*, 4670:449–450, 2007.

[39] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir. An introduction to fuzzy answer set programming. *Ann. Math. Artif. Intell.*, 50(3-4):363–388, 2007.

[40] G. Wagner. Negation in fuzzy and possibilistic logic programs. In T. P. Martin and F. A. Fontana, editors, *Logic programming and soft computing*, chapter 6, pages 113–128. Taylor & Francis, 1998.

[41] G. Wagner. Web rules need two kinds of negation. *Lecture Notes in Computer Science*, 2901:33–50, 2003.

[42] P. Wong and P. Besnard. Paraconsistent reasoning as an analytic tool. *Logic Journal of the IGPL*, 9(2):217–229, 2001.

APPENDIX
PROOFS OF THE RESULTS

Proposition 1

Proof of proposition 1: It is known that if $(*, \leftarrow)$ forms an adjoint pair, then $*$ has to be monotonic in both arguments (see [12]). Then the monotonicity of $T_{\mathbb{P}}$ is straightforward since each operator in its definition is monotonic. ∎

Proposition 2

Proof: We show firstly that if an L-interpretation I is coherent then every L-interpretation J such that $J \leq I$ is coherent as well. Let p be a propositional symbol, then by the inequality $I \leq J$, the coherence of J and the decreasing property of $\sim$ we have

$$I(\sim p) \leq J(\sim p) \leq \dot{\sim} J(p) \leq \dot{\sim} I(p)$$

Therefore, if there is a coherent model of $\mathbb{P}$, the least model has to be necessarily coherent as well. ∎

Lemma 1

Proof: If M is a stable model of $\mathbb{P}$ then M is a model of $\mathbb{P}_M$, this is, for every rule $\langle \ell \leftarrow \mathcal{B}^+ \ ; \ M(\mathcal{B})^- * \vartheta \rangle$ in $\mathbb{P}_M$, the following inequality holds

$$M(\ell) \geq M(\mathcal{B}^+) * M(\mathcal{B}^-) * \vartheta$$

but this is equivalent to say that M is a model of $\mathbb{P}$. ∎

Theorem 1

Proof: Let us prove in advance that each model N of $\mathbb{P}$ is a model of $\mathbb{P}_N$ as well. Each rule in $\mathbb{P}_N$, with the form $r_N \colon \langle \ell \leftarrow \mathcal{B}^+ \ ; \ N(\mathcal{B})^- * \vartheta \rangle$, comes from a rule in $\mathbb{P}$ with the form $r \colon \langle \ell \leftarrow \mathcal{B}^+ * \mathcal{B}^- \ ; \vartheta \rangle$. As N satisfies r, the following inequality holds:

$$N(\ell) \geq N(\mathcal{B}^+) * N(\mathcal{B}^-) * \vartheta$$

which is equivalent to say that N satisfies r_N. Therefore N is a model of $\mathbb{P}_N$

On the other hand, if M is a stable model of $\mathbb{P}$ and $N \subseteq M$ is a model of $\mathbb{P}$ then, by the reasoning above, N is a model of $\mathbb{P}_N$; by using that $\dot{\neg}$ is a decreasing operator, we obtain the following inequality for each rule $r \colon \langle \ell \leftarrow \mathcal{B}^+ * \mathcal{B}^- \ ; \vartheta \rangle$ in $\mathbb{P}$:

$$N(\ell) \geq N(\mathcal{B}^+) * N(\mathcal{B}^-) * \vartheta \geq N(\mathcal{B}^+) * M(\mathcal{B}^-) * \vartheta$$

which implies that N is a model of $\mathbb{P}_M$. As M is the least model of $\mathbb{P}_M$ and $N \subseteq M$, we conclude that $N = M$. ∎

Proposition 3

Proof: The proof is direct from the definition of $\mathcal{I}(p; I)$. ∎

Proposition 4

Proof: As L is finite, the infimum is actually a minimum. Therefore, there is a coherent pair (x, y) such that $0 = \mathcal{I}(p; I) = m(I(p)) - m(x) + m(I(\sim p)) - m(y)$. Let us see that $m(I(p)) - m(x) = 0$ and $m(I(\sim p)) - m(y) = 0$. If $m(I(p)) - m(x)$ were greater strictly than 0 then $m(I(\sim p)) - m(y)$ should be lesser strictly than 0; but that is imposible since $m(I(\sim p)) \geq m(y)$. Hence, as m is injective, $I(p) = x$ and $I(\sim p) = y$ hold necessarily. Therefore, $(I(p), I(\sim p))$ is a coherent pair or, in other words, p is a coherent symbol. ∎

Proposition 5

Proof: The proof comes from the fact that $\sim$ continuous implies that $\Delta_I^{\sim}(p)$ is a compact. Therefore, as m is continuous, $\mathcal{I}(p; I)$ is actually a minimum. The proof follows as that of Proposition 4. ∎

Proposition 6

Proof: It suffices to prove that

$$m(J(p)) - m(x) + m(J(\sim p)) - m(y) \geq \mathcal{I}(I; p)$$

for all $(x, y) \in \Delta_J^{\sim}(p)$.

Firstly, note that $(\inf\{x, I(p)\}, \inf\{y, I(\sim p)\}) \in \Delta_I^{\sim}(p)$, since the following inequalities hold

$$\begin{aligned} \inf\{x, I(p)\} &\leq I(p) \\ \inf\{y, I(\sim p)\} &\leq I(\sim p) \\ \inf\{y, I(\sim p)\} \leq y &\leq \dot{\sim} x \leq \dot{\sim} \inf\{x, I(p)\} \end{aligned}$$

Now, consider the following chain of inequalities:

$$\begin{aligned} m(J(p)) - m(x) + m(J(\sim p)) - m(y) &\overset{(a)}{\geq} \\ m(\sup\{x, I(p)\}) - m(x) + m(\sup\{y, I(\sim p)\}) - m(y) &\overset{(b)}{\geq} \\ m(I(p)) - m(\inf\{x, I(p)\}) + m(I(\sim p)) - m(\inf\{y, I(\sim p)\}) & \end{aligned}$$

where inequality (a) follows because of monotonicity, and inequality (b) is a consequence of the third property of information measures.

Finally, note that the last line in the previous chain is a particular case of the elements involved in the definition of $\mathcal{I}(I; p)$ and, hence, is greater than or equal to it.

∎

Proposition 7

Proof: Note that $(0, I(\sim p)) \in \Delta_I^{\sim}(p)$. Therefore, $\mathcal{I}(p, I) \leq m(I(p)) - m(0) + m(I(\sim p)) - m(I(\sim p)) = m(I(p))$. Similarly, we obtain $\mathcal{I}(p, I) \leq m(I(\sim p))$. ∎

Proposition 8

Proof: For $i = 1$ the proof follows the same line than that of Proposition 2, i.e. if p is coherent w.r.t. J, then p is coherent w.r.t. I. As a result, every incoherent propositional symbol w.r.t. I is also incoherent w.r.t. J.

For $i = 2, 3$ the proof follows from Proposition 6. ∎

Proposition 9

Proof: As $\mathbb{P}$ and $\mathbb{Q}$ are extended logic programs, both have a unique stable model, which coincides with the respective least model. We will show first that $\text{lfp}(T_{\mathbb{P}}) \leq \text{lfp}(T_{\mathbb{Q}})$

For every interpretation I and literal ℓ, the following inequality holds

$$\begin{aligned} T_{\mathbb{P}}(I)(\ell) = \sup\{I(\mathcal{B}) * \vartheta \colon \langle \ell \leftarrow \mathcal{B}; \quad \vartheta \rangle \in \mathbb{P}\} \leq \\ \leq \sup\{I(\mathcal{B}) * \overline{\vartheta} \colon \langle \ell \leftarrow \mathcal{B}; \quad \overline{\vartheta} \rangle \in \mathbb{Q}\} \leq T_{\mathbb{Q}}(I)(\ell) \end{aligned}$$

by monotonicity of $*$ and the inequality $\mathbb{P} \leq \mathbb{Q}$. As a consequence, $\text{lfp}(T_{\mathbb{P}}) \leq \text{lfp}(T_{\mathbb{Q}})$ holds.

Now, by Proposition 8 we would have the following inequality, for all $i = 1, 2, 3$:

$$\mathcal{I}_i(\mathbb{P}) = \mathcal{I}_i(\text{lfp}(T_{\mathbb{P}})) \leq \mathcal{I}_i(\text{lfp}(T_{\mathbb{Q}})) = \mathcal{I}_i(\mathbb{Q})$$

∎

Proposition 10

Proof: Let us consider $X = \{r_1, \ldots, r_k\}$ and $Y = X \cup \{r_{k+1}, \ldots, r_n\}$. For each tuple $\{\varphi_i\}_{i=1,\ldots,k}$ such that $\Theta_{\mathbb{P}}(X, \{\varphi_i\}_i)$ is stable, we consider the tuple $\{\psi_j\}_{j=1,\ldots,n}$ defined by $\psi_j = \varphi_j$ for $j = 1, \ldots, k$ and $\psi_j = 0$ for $j = k+1, \ldots, n$. Obviously $\Theta_{\mathbb{P}}(Y, \{\psi_j\}_j)$ is stable as well and:

$$\sum_{i=1}^{k} \Big(m(1) - m(\varphi_i)\Big) = \sum_{i=1}^{n} \Big(m(1) - m(\psi_i)\Big)$$

The inequality $\textsc{Inst}^-_{\mathbb{P}}(X) \geq \textsc{Inst}^-_{\mathbb{P}}(Y)$ is now clear by definition of infimum. ∎

Corollary 2

Proof: It suffices to show that the set on which the infima are computed in the definition of $\textsc{Inst}^-_{\mathbb{P}}(\mathbb{P})$ and $\textsc{Inst}^+_{\mathbb{P}}(\mathbb{P})$ are non-empty.

For $\textsc{Inst}^-_{\mathbb{P}}(\mathbb{P})$, simply note that $\Theta_{\mathbb{P}}(\mathbb{P}, \{0\}_i)$ is always stable since is the empty program. On the other hand, for $\textsc{Inst}^+_{\mathbb{P}}(\mathbb{P})$ we have that $\Omega_{\overline{\mathbb{P}}}(\overline{\mathbb{P}}, \{1\}_i)$ is always stable since the interpretation $M(\ell) = 1$ for all $\ell \in Lit_{\mathbb{P}}$ is its unique stable model. Therefore, the sets used to determine $\textsc{Inst}^-_{\mathbb{P}}(\mathbb{P})$ and $\textsc{Inst}^+_{\mathbb{P}}(\overline{\mathbb{P}})$ are non-empty and thus the infima exist.

Both $\textsc{Inst}^-_{\mathbb{P}}(X) \leq \textsc{Inst}^-_{\mathbb{P}}(\mathbb{P})$ and $\textsc{Inst}^+_{\mathbb{P}}(X) \leq \textsc{Inst}^+_{\mathbb{P}}(\overline{\mathbb{P}})$ hold as a consequence of Proposition 10. ∎

Proposition 11

Proof: The first item follows from the definition of $\textsc{Inst}^-_{\mathbb{P}}(X)$ and $\textsc{Inst}^+_{\mathbb{P}}(X)$. The second and third items are consequence of the definition of infimum in the real numbers. ∎

Lemma 2

Proof: Assume that N is a model of $(\mathbb{P}^+_X)_M$. Every rule r_j in $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$ whose weight has not been changed by the operator $\Omega_{\overline{\mathbb{P}}}$ is trivially satisfied by N, since r_j belongs to $(\mathbb{P}^+_X)_M$ as well; thus, we only need to consider the case of rules $\langle r_i; s(\vartheta_i, M(\alpha_i))\rangle_M$ such that $\langle r_i; \vartheta_i\rangle \in X$.

As N satisfies the reduct w.r.t. M of rule (9) associated to r_i (included in the construction of $\mathbb{P}^+_X$), then we have:

$$N(\ell_i) \geq N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * \mathcal{N}(M(\gamma_i))$$

Now, substituting the value of $M(\gamma_i)$:

$$\begin{array}{rcl} N(\ell_i) & \geq & N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * \mathcal{N}(t_s(\mathcal{N}(\vartheta_i), \mathcal{N}(M(\alpha_i)))) \\ & = & N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * s(\vartheta_i, M(\alpha_i)) \end{array}$$

Thus, N satisfies $\langle r_i; s(\vartheta_i, M(\alpha_i))\rangle_M \in \Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$.

Reciprocally, assume now that N is a model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$. For each rule r_j in $(\mathbb{P}^+_X)_M$ not corresponding to the reduct of some of the rules included by (5, (6), (9), and (10), the proof is straightforward, since r_j belongs to $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_{M|_{\Pi_{\mathbb{P}}}}$ as well. Otherwise, the interpretation N has to be extended, and the values of $N(\beta_i)$ and $N(\gamma_i)$ are defined by using $M(\alpha_i)$ as follows:

$$N(\beta_i) = \mathcal{N}(M(\alpha_i)) \qquad N(\gamma_i) = t_s(\mathcal{N}(\vartheta_i), \mathcal{N}(M(\alpha_i)))$$

Now, the reducts of rules corresponding to (5), (6), and (10) are satisfied; only the satisfiability of the reduct of the rules of type (9) need some calculations. As N satisfies the reduct of the rule $\langle r_i; s(\vartheta_i, M(\alpha_i))\rangle$, the following inequality holds:

$$N(p_i) \geq N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * s(\vartheta_i, M(\alpha_i))$$

Using the equality $s(x, y) = \mathcal{N}(t_s(\mathcal{N}(x), \mathcal{N}(y)))$:

$$\begin{array}{rcl} M(\ell_i) & \geq & N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * s(\vartheta_i, M(\alpha_i)) \\ & = & N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * \mathcal{N}(t_s(\mathcal{N}(\vartheta), \mathcal{N}(M(\alpha_i)))) \\ & = & N(\mathcal{B}^+_i) * M(\mathcal{B}^-_i) * \mathcal{N}(M(\gamma_i)) \end{array}$$

which proves that the rule is satisfied by N. ∎

Proposition 12

Proof: Let M be a stable model of $\mathbb{P}^+_X$, then M is the least model of $(\mathbb{P}^+_X)_M$. For each β_i, the value $M(\beta_i)$ has to be necessarily $\mathcal{N}(M(\alpha_i))$ since the only rule whose head is β_i in $(\mathbb{P}^+_X)_M$ is:

$$\langle \beta_i \leftarrow \quad ; \mathcal{N}(M(\alpha_i))\rangle$$

Similarly, for each γ_i we can infer that $M(\gamma_i) = t_s(\mathcal{N}(\vartheta_i), \mathcal{N}(M(\alpha)))$, therefore the hypotheses of Lemma 2 hold, and M is a model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i\}))_M$ as well.

We can see that $M|_{Lit_{\mathbb{P}}}$ is a stable model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})$ since if $N \subset M|_{Lit_{\mathbb{P}}}$ were a model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_{M|_{Lit_{\mathbb{P}}}}$ then, by Lemma 2, N could be extended to a model of $(\mathbb{P}^+_X)_M$ satisfying $N \subset M$; which is a contradiction with the minimality of M.

Reciprocally, let M be a stable model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})$, that is, M is the least model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$. Therefore, if we extend M to β_i and γ_i via $M(\beta_i) = \mathcal{N}(M(\alpha))$ and $M(\gamma_i) = t_s(\mathcal{N}(\vartheta_i), \mathcal{N}(M(\alpha_i)))$, Lemma 2 can be applied once again and, thus M is also a model of $(\mathbb{P}^+_X)_M$.

If M were not the least model of $(\mathbb{P}^+_X)_M$ then, there would exist a model N of $(\mathbb{P}^+_X)_M$ such that $N \subset M$. By using Lemma 2 again, $N \subset M$ would be a model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})_M$; and this would contradict the fact that M is a stable model of $\Omega_{\overline{\mathbb{P}}}(X, \{M(\alpha_i)\})$.

∎

Nicolás Madrid received the M.Sc. degree in Mathematics from the University of Málaga, Spain, in 2006. In 2007, he received a pre-doctoral grant by the Spanish Ministry of Science and Innovation to conduct research for a four years period with the Department of Applied Mathematics, University of Málaga. He is currently preparing his Ph.D. His current research interests include fuzzy answer set semantics and dealing with inconsistencies.

Prof. Manuel Ojeda-Aciego received the M.Sc. in Mathematics in 1990, and the Ph.D. in Computer Science in 1996, both from the University of Málaga, Spain. He is currently a Professor in the Department of Applied Mathematics, University of Málaga, and has authored or coauthored more than 100 papers in scientific journals and proceedings of international conferences. He has co-edited the book Foundations of Reasoning under Uncertainty (Springer-Verlag, 2010), as well as several special issues in scientific journals on mathematical and logical foundations of non-classical reasoning. His current research interests include fuzzy answer set semantics, residuated and multi-adjoint logic programming, fuzzy formal concept analysis, and algebraic structures for computer science. He is president of the Computer Science Committee of the Royal Spanish Mathematical Society, member of EUSFLAT, and senior member of the IEEE.